TAMPER-RESILIENT METHODS FOR WEB-BASED OPEN SYSTEMS

A Thesis
Presented to
The Academic Faculty

by

James Caverlee

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Computing

Georgia Institute of Technology
August 2007

TAMPER-RESILIENT METHODS FOR WEB-BASED OPEN SYSTEMS

Approved by:

Ling Liu, Advisor
College of Computing
Georgia Institute of Technology

William B. Rouse, Co-Advisor
Tennenbaum Institute, Industrial & Systems Engineering (Joint Appointment with College of Computing)
Georgia Institute of Technology

Jonathon Giffin
College of Computing
Georgia Institute of Technology

Sham Navathe
College of Computing
Georgia Institute of Technology

Sushil K. Prasad
Department of Computer Science
Georgia State University

Calton Pu
College of Computing
Georgia Institute of Technology

Date Approved: 18 June 2007

For Sherry

ACKNOWLEDGEMENTS

I am pleased to acknowledge the generous support of the Tennenbaum Institute in providing me a Tennenbaum Fellowship for this research. In addition, I am grateful for the positive influence and the many contributions of a number of special people – people who have encouraged, guided, and inspired me.

My parents are my foundation. Sam and Ellen Caverlee are two incredibly encouraging and supportive parents. They have been with me every step of the way, encouraging me to push myself to my limits and supporting me when I fall short. They are an inspiration to me and provide a blueprint for how to live a moral life. I love you, Mom and Dad. My brother John has always been there for me. John has a way of grounding me, so that I maintain my focus and don't float off into a self-imposed nerd orbit. My grandfather Jim is a constant source of inspiration. The rest of my family is equally amazing. I am fortunate to have such a great family.

When I first met Ling Liu in the fall of 2002, I knew I had met someone with a fierce intellect and strong opinions. As I have grown to know her, I have seen firsthand her generous spirit and tireless work ethic. Ling has had the single greatest impact on my intellectual and professional development, and she deserves much of the credit for leading me on this journey. Ling is an inspiration to me, and I hope to guide my students as well as she has guided me. I am especially grateful to Bill Rouse. Bill is thoughtful and engaging, and it has been a real pleasure to participate in the growth of the Tennenbaum Institute. Bill has been a model for my professional development, and I would be happy to achieve half of what he has. I am thankful to Calton Pu, another fine model for my professional development. Calton has always offered welcome advice, as well as encouraged me along the way. Leo Mark has been a great source of support from my first days at Georgia Tech. His enthusiasm has been a great motivator.

I am indebted to Ling, Bill, Calton, Jonathon Giffin, Sham Navathe, and Sushil K.

Prasad for reading and commenting on my thesis. Dave Buttler and Dan Rocco welcomed me to Georgia Tech and showed me what it takes to be a graduate student. Li Xiong and Steve Webb were both great officemates and are now my good friends. I have had the great opportunity to collaborate with a number of really smart and interesting characters in my time at Georgia Tech. I am especially grateful to the members of the Distributed Data Intensive Systems Lab. Thanks to Susie McClain and Deborah Mitchell for helping me find my way in the maze that is graduate school. Outside of school, I have been fortunate to have a wonderful group of friends at Marietta First United Methodist Church.

Sometimes I think my wife Sherry is a superhero, only without the cape. She is an amazing person, a source of boundless love and energy, and my best friend. She has endured much because of me. I am a better man because of her. Thank you, sweetie.

Finally, this acknowledgment would not be complete without a nod to my two beautiful children. Luke and Eva, you are an inspiration. Know that I love you always.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The Web and Web-based open systems are characterized by their massive amount of data and services for leveraging this data. These systems are noted for their open and unregulated nature, self-supervision, and high degree of dynamism, which are key features in supporting a rich set of opportunities for information sharing, discovery, and commerce. But these open and self-managing features also carry risks and raise growing concerns over the security and privacy of these systems, including issues like spam, denial-of-service, and impersonated digital identities.

Our focus in this thesis is on the design, implementation, and analysis of large-scale Web-based open systems, with an eye toward enabling new avenues of information discovery and ensuring robustness in the presence of malicious participants. We identify three classes of vulnerabilities that threaten these systems: vulnerabilities in link-based search services, vulnerabilities in reputation-based trust services over online communities, and vulnerabilities in Web categorization and integration services. This thesis introduces a suite of methods for increasing the tamper-resilience of Web-based open systems in the face of a large and growing number of threats. We make three unique contributions:

- First, we present a source-centric architecture and a set of techniques for providing tamper-resilient link analysis of the World Wide Web. We propose the concept of link credibility and present a credibility-based link analysis model. We show that these approaches significantly reduce the impact of malicious spammers on Web rankings.

- Second, we develop a social network trust aggregation framework for supporting tamper-resilient trust establishment in online social networks. These community-based social networking systems are already extremely important and growing rapidly. We show that our trust framework supports high quality information discovery and is robust to the presence of malicious participants in the social network.

- Finally, we introduce a set of techniques for reducing the opportunities of attackers to corrupt Web-based categorization and integration services, which are especially important for organizing and making accessible the large body of Web-enabled databases on the Deep Web that are beyond the reach of traditional Web search engines. We show that these techniques reduce the impact of poor quality or intentionally misleading resources and support personalized Web resource discovery.

CHAPTER I

INTRODUCTION

The explosive rise of the Web and Web-based open systems has had a profound transformative impact on knowledge discovery and dissemination, business strategy and commerce, and even the structure of our interpersonal relationships. The growing reliance on the Web and Web-based open systems - including online social networks (like Facebook and MySpace), social media and search services (like YouTube, Digg, and Wikipedia), online vendors (like Amazon and Netflix), and blogspaces (like Blogger and Livejournal) – present a number of great opportunities and unique challenges.

Consider the Web. From its infancy in the early 1990s, the Web has grown from 200 million indexable pages in 1997 [21] to more than 11.5 billion pages in 2005 [76]. Coupled with data in Web-enabled databases and other dynamic data sources and the effective size of the Web is staggering [86, 117]. With this massive growth, we have witnessed tremendous research and commercial strides in organizing, sorting, and understanding the Web's massive amount of data. These advances have encouraged the emergence of innovative new companies and business strategies, permanently changing how businesses operate and the quality of our daily lives. Indeed, the ubiquitous impact of the Web has spurred calls for a new discipline studying the science of the Web [18, 19]. As the Web evolves to provide the information infrastructure for pervasive and mobile devices, we can anticipate many exciting new developments.

Similarly, online communities have grown dramatically in the past few years, and are increasingly recognized as important enablers of information sharing, electronic commerce, and new modes of social interaction. Social networking services like the ones offered by MySpace and Facebook support the management of social relationships, connecting millions of users. MySpace alone has grown from 1 million user accounts in the first quarter of 2004

to an astonishing 190 million user accounts today.[1] Similarly, community-powered services like the online encyclopedia Wikipedia, the image-sharing site Flickr, the community news portal Digg, and the video-sharing site YouTube have flourished as users have explored and contributed to these community-based information spaces.

While the growth of the Web and these online communities has led to increasing media attention and popular awareness, there has been a commensurate rise in concern over their security and privacy. For example, recent studies have indicated that a significant fraction of Web pages contain either executables bundled with spyware [130, 142] or zero-day browser exploits [171] that can be used to compromise a user's computer. The large social networking sites have been the target of specialized phishing attacks and massive email spam campaigns that leverage the wealth of personal information and relationship data that may be mined from these communities [42, 94]. Malicious users have been caught impersonating others: e.g., in April 2006, an 18-year-old was caught impersonating a teacher on MySpace; the impersonator made racist statements and falsely represented the teacher as a pornographer and child molester [155]. Other threats have been observed – including the massive Samy worm that overloaded MySpace with 1 million friend requests in a single day [101]– and new threats are certain to emerge as attackers grow in sophistication.

Web publishing is fairly inexpensive and there are minimal hurdles to limit who can or cannot participate on the Web. Similarly, most of the online communities are relatively open, typically requiring only a simple user registration or a valid email address. This openness has helped drive the massive growth of the Web and online communities, but the observed security and privacy problems question the viability and safety of such an open approach as more and more malicious entities target them.

1.1 Research Challenges

Building effective open Web-based open systems is challenging, and we believe that significant research advances are needed to support their continued use and to encourage their adoption by government and business entities interested in providing more effective and

[1] `www.myspace.com`

responsive governance, customer relationship management, emergency and disaster responsiveness, and other mission-critical services. As more and more malicious entities target these systems, what guarantees can we make? Can we develop algorithms and architectures for the efficient, reliable, and secure use of the Web? Can we build systems in which users have trust in the data and information derived from these systems, even in the presence of users intent on undermining the quality of information? In particular, we identify several research challenges in building tamper-resilience into Web-based open systems:

- **Openness:** The Web is inherently open in terms of who can access it, who can publish to it, and how data is shared, though it has some restrictions in terms of the languages and protocols it respects (e.g., HTML, HTTP). Maintaining this openness is important for its continued growth and impact. Can we provide some regulation while maintaining the degree of openness such that we can provide some degree of privacy and security requirements typically found in tightly controlled systems?

- **Accessibility:** Any regulation of the Web should maintain the accessibility of data and information on the Web, without severely restricting what can and cannot be discovered. Can the tamper-resilience methods developed assure this continued accessibility? Can we provide high assurance of the accuracy and quality of these systems in the face of a growing number of security and privacy threats?

- **Scalability:** The Web is very large and continues to grow and to expand into new forms, with the emergence of pervasive and mobile devices for accessing the Web and massive eScience and grid initiatives for managing large distributed computing applications. Techniques for tamper-resilience should respect this scalability.

- **Personalization:** Seemingly at odds with scalability is personalization. Users have different perspectives on what is and what is not interesting and relevant. Can we develop algorithms and architectures that support a customizable personalized view of the Web?

Naturally, when building tamper-resilience in open systems, we are faced with the classic arms race cycle, that is – (i) a solution is proposed; (ii) the adversaries adapt their techniques to subvert the solution; (iii) the solution is revised, the adversaries adapt, and the cycle continues. In this thesis, we are interested in exploring whether we can *significantly raise the costs* to adversaries, so that they wield only a *limited* ability to respond to the proposed countermeasures and to continue the arms race cycle.

1.2 Overview of Thesis

This thesis is concerned with vulnerabilities in the principles, design, and maintenance of massive Web-based open systems. We argue that tamper-resilience must be built-in to preserve the quality of these open systems against malicious manipulation and accidental misuse. Our goal is to develop effective tamper-resilient methods for supporting Web-based open systems.

In particular, we focus on the design, implementation, and analysis of large-scale Web-based open systems, with an eye toward enabling new avenues of information discovery and ensuring robustness in the presence of malicious participants. We identify three classes of vulnerabilities that threaten these systems: vulnerabilities in link-based search services, vulnerabilities in reputation-based trust services over online communities, and vulnerabilities in Web categorization and integration services. To address these risks, we introduce a suite of methods for increasing the tamper-resilience of Web-based open systems in the face of a large and growing number of threats. One of the goals in this thesis is to study specific problems, but also consider how the lessons and principles learned can be applied in other problem domains.

In particular, this thesis research makes three unique contributions toward this direction:

- First, we present a source-centric architecture and a set of techniques for providing tamper-resilient link analysis of the World Wide Web. We propose the concept of link credibility and present a credibility-based link analysis model. We show that these approaches significantly reduce the impact of malicious spammers on Web rankings.

- Second, we develop a social network trust aggregation framework for supporting

tamper-resilient trust establishment in online social networks. These community-based social networking systems are already extremely important and growing rapidly. We show that our trust framework supports high quality information discovery and is robust to the presence of malicious participants in the social network.

- Finally, we introduce a set of techniques for reducing the opportunities of attackers to corrupt Web-based categorization and integration services, which are especially important for organizing and making accessible the large body of Web-enabled databases on the Deep Web that are beyond the reach of traditional Web search engines. We show that these techniques reduce the impact of poor quality or intentionally misleading resources and support personalized Web resource discovery.

Our methods for increased tamper-resilience are complementary to traditional information security approaches. Traditional methods address the authentication, authorization, confidentiality, and integrity of the users and messages. Concretely, cryptographic techniques are used for authenticating the identity of users and securing the publishing infrastructure, or digital signatures are used for providing information integrity, or access control techniques are used for constraining user behavior. Our work complements these traditional methods by addressing manipulation and attacks against the intelligence of these systems, including the underlying principles, usage, and maintenance of these systems.

The rest of this thesis is organized as follows:

- **Chapter 2: Web Risks and Vulnerabilities** – We begin with an overview of the numerous Web risks and vulnerabilities that have been observed and experienced in Web-based open systems. We provide a taxonomy of risks and vulnerabilities, illustrate each with a concrete example, and position our contributions with respect to this taxonomy.

- **Chapter 3: Source-Centric Web Ranking** – Since ranking systems (like those offered by search engines) play a central role in organizing the Web for the vast majority of Web users, Web spammers spend a considerable effort on manipulating the

underlying algorithms that drive these ranking systems. In this chapter, we introduce a source-centric model for Web ranking that promotes a hierarchical abstraction of the Web graph based on the strong Web link structure. Complementary to the traditional page-based view of the Web, our source approach naturally limits spam opportunities and incorporates a novel notion of influence throttling for countering the influence of spammers.

- **Chapter 4: Credibility-Based Link Analysis** – In this chapter, we advocate a clean separation of Web resource quality and link quality and argue that the intrinsic quality of a Web resource should be distinguished from its intrinsic link credibility. We formally define the concept of credibility, study the factors impacting the computation of credibility, and introduce a novel credibility-based Web ranking algorithm, which incorporates credibility information directly into the quality assessment of each Web resource. We show that the credibility-based approach is significantly and consistently more spam-resilient than the state-of-the-art.

- **Chapter 5: Trust Establishment in Online Communities** – In this chapter, we present the design and evaluation of the SocialTrust framework for aggregating trust in online communities. The framework supports tamper-resilient trust establishment through four key features – (i) trust establishment scope; (ii) trust group feedback; (iii) relationship link quality; and (iv) the history of the user's behavior in the community. We find that SocialTrust supports robust trust establishment even in the presence of large-scale collusion by malicious participants.

- **Chapter 6: Controlled Sampling of Web Resources** – Web categorization and integration services for providing access to the Deep Web may be corrupted by Web spammers in an effort to disrupt these services and hurt the quality of the information provided or to drive Web users to spam partners for online commerce and to support identity theft. In this chapter, we present a technique for controlled sampling of Web resources. The controlled sampling architecture supports the extraction of high-quality database samples for reducing the impact of poor quality or intentionally

misleading resources.

- **Chapter 7: Trusted Web Resource Discovery** – To support tamper-resilient categorization and integration, we present the design and implementation of a Web resource discovery system. By leveraging a set of user-trusted information resources, the system provides a set of related information resources that can be grouped by content and trustworthiness properties to auto-generate a trusted categorization hierarchy, supporting a personalized view over a collection of Deep Web databases through source-biased database analysis and exploration. We have developed a prototype system that is designed for crawling, probing, and supporting personalized Deep Web databases discovery using the source-biased approach.

- **Chapter 8: Conclusions and Future Work** – We conclude with a summary of our thesis contributions and a discussion of future research extensions to the results presented here.

CHAPTER II

WEB RISKS AND VULNERABILITIES

The Web continues to be the largest and most frequently accessed living data repository. As the Web has grown and increasingly become the primary portal for sharing information and supporting online commerce, there has been a rise in efforts to pollute the Web with low-quality and dangerous Web content and to manipulate how users view and interact with the Web. With the increasing reliance on Web-based marketplaces, social networks, and large-scale information sharing communities, individuals and their computer systems are at risk for abuse and exploitation at the hands of malicious actors. These vulnerabilities degrade the quality of information on the Web and place the user at risk for exploitation.

In this chapter, we survey the current state-of-the-art with respect to (i) content-based Web risks; (ii) experience-based Web risks; and (iii) discuss some related work that attempts to deal with this growing threat. This survey is intended to provide an overview of the current situation, and is not intended to be exhaustive.

2.1 Content-Based Web Risks

We begin by examining some of the currently observed dangerous Web content. Web publishing has become increasingly cheap and simple, and there are many turn-key solutions available for easy Web site deployment. As a result, the barriers to making dangerous Web content available to a wide audience are correspondingly low. We shall refer to a Web publisher who deploys or supports such dangerous Web content as a *spammer*.

2.1.1 Badware

One of the most concerning examples of dangerous Web content is badware. We use the term badware to refer to any type of malicious software on the Web. Examples include keyloggers designed for user tracking, adware that pops-up annoying advertisements, as well as viruses and trojans that are part of a Web-borne attack vector (e.g., for commanding a botnet army

Figure 1: Site That Bundles Badware with Downloadable Games

[67]). A user can be infected with badware through a number of avenues. Badware may be bundled with a seemingly innocent downloaded executable. In Figure 1 we show a sample Web site that hosts downloadable games and screensavers, but also surreptitiously installs an adware program when the user downloads a game. These sites often have a "look-and-feel" that is similar to legitimate sites, making it difficult to detect the deception. Pop-up browser windows that emulate legitimate operating system prompts, like the one in Figure 2, can fool users into downloading badware. One recent study found that as of October 2005, nearly 5% of all executables discovered on the Web were bundled with some form of badware [130]. Perhaps even more troubling, malicious Web publishers can insert code into their Web pages that can directly compromise the Web browser, without any action on the part of the user. In these instances of "drive-by downloads" a user need only view a compromised page to be infected. One recent Google study identified 450,000 unique URLs that engaged in this form of attack and raised the alarming concern that this attack vector could support botnet formation and deployment [142]. Similarly, a Microsoft study identified a number of Web-hosted zero-day Windows exploits – these exploits could compromise a fully-patched and updated Windows machine [171].

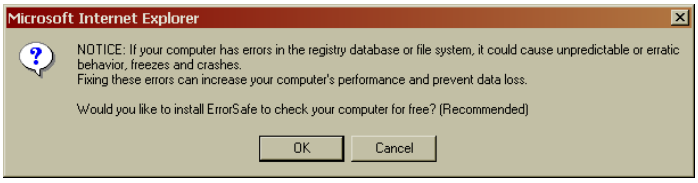

Figure 2: Pop-Up Window For Downloading Badware

2.1.2 Page Spoofing

The second Web risk we identify is page spoofing (or page forgery). In this case, a spammer pollutes the Web with clones of legitimate sites (like eBay). In doing so, the spammer can distribute badware or engage in identity theft, since the user believes that she is engaging with a legitimate Web site. Recent estimates suggest that at least two million Web users have voluntarily divulged private information to spoofed Web sites, resulting in $1.2 billion in losses for banks and credit card issuers [51]. In Figure 3, we illustrate a spoofed Ameritrade site; the critical clue that this is not the legitimate Ameritrade site is the non-standard URL. Spoofed pages are often used in concert with social engineering techniques to lure users to these sites, for example, through a phishing email that includes a URL to the fake page.[1]

2.1.3 Offensive Content

Instead of spreading badware or stealing a user's identity, some Web content may simply be illegal or offensive to a user. Some Web content is considered universally illegal – like child pornography. Other content may be illegal in one jurisdiction but not another – like Web-based gambling sites that are intended to be off-limits to U.S. citizens and so are often hosted offshore [5], or hate speech in certain European countries [157]. And then there may be some content that is perfectly legal, but of offense to a particular user or group of users. For example, there is great evidence that many families prefer to install filters for safeguarding their children from certain types of Web content, be it pornography, drugs, or extreme violence. Naturally, what is and what is not offensive is debateable, and there are arguments for and against Web freedom-of-expression versus censorship.

[1] `http://www.antiphishing.org/`

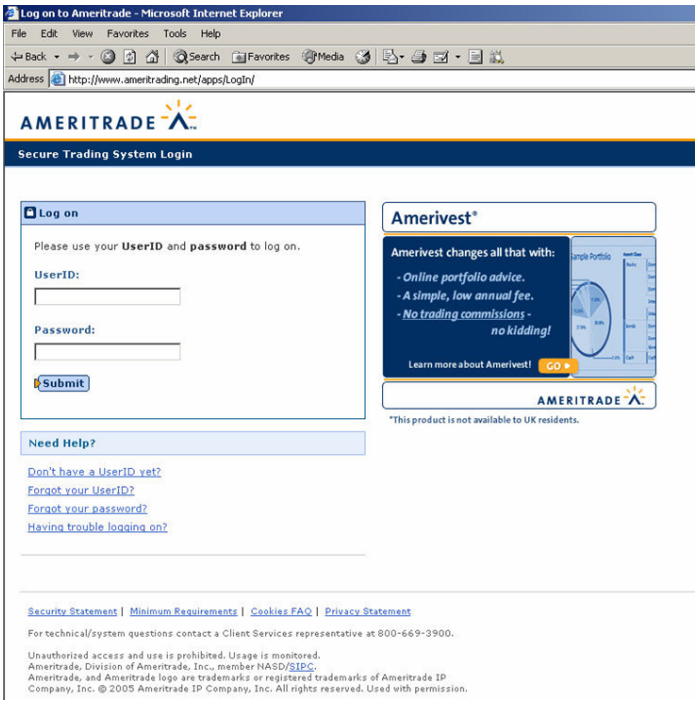

Figure 3: Page Spoofing Example: Ameritrade

2.1.4 Useless or Derivative Content

Unlike offensive content, some Web content is merely useless or derivative. A number of studies have identified a large portion of the Web to be duplicated from other portions of the Web. This duplication occurs on a page-level [41, 60], where entire Web pages are duplicated, down to a phrase-level, where particular passages from pages are copied [31, 62]. The motivation for the duplication can be innocuous – e.g., identical AP news stories may be published by separate news sites and some technical FAQs are published in many places. But in some cases duplicate content is intended to deceive; for example, Figure 4 illustrates a casino guide that has duplicated some Wikipedia content; at the bottom of the page, spam hyperlinks to pill sites are inserted. Similarly, spammers "weave" together content scraped from other sites [79] or auto-generate Web sites and blogs that pollute the Web with low-quality content. Figure 5 is an example of a machine-generated blog about Brandon Routh, who portrayed Superman in the recent film. The text of the blog is nonsensical, though tangentially related to Superman (e.g., "romantic, action-packed Oneida Dispatch, NY - 28 June 2006 The Man of Steel was"). In the following section, we will discuss some

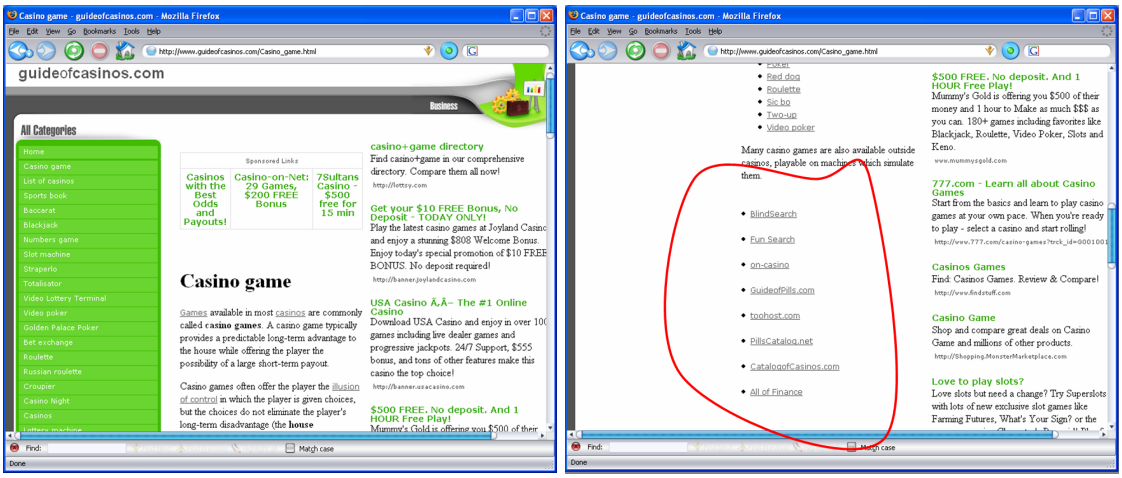

Figure 4: Site With Duplicate Wikipedia Content and Spam Links to Pill Sites

of the reasons why spammers generate such useless or derivative content.

2.1.5 Propaganda

Since the Web is often the first and last resort of information for some users, Web publishers have an incentive to shape how users view the world around them. We see evidence of this behavior with respect to consumer products, restaurants, and other goods and services. Online commerce sites like eBay and Amazon that aggregate user reviews and ratings to make product recommendations are subject to manipulation [104]. In one instance, book authors were caught on Amazon writing rave reviews of their own books (using a pseudonym) and negatively reviewing the books of their competitors [81]. In a similar fashion, Microsoft has been accused of paying third-parties to portray their products in a more favorable light on Wikipedia [17]. For shaping one's social standing on the Web, several companies offer tools for automatically generating friends on the popular social networking sites (e.g., `www.friendbot.com`).

2.1.6 Discussion

Dealing with content-based Web risks is an open and active area of research and commercial interest. For identifying Web sites that host malicious code, Microsoft has developed a custom Web crawler as part of the Strider HoneyMonkey Project.[2] Company offerings like

[2] `http://research.microsoft.com/HoneyMonkey/`

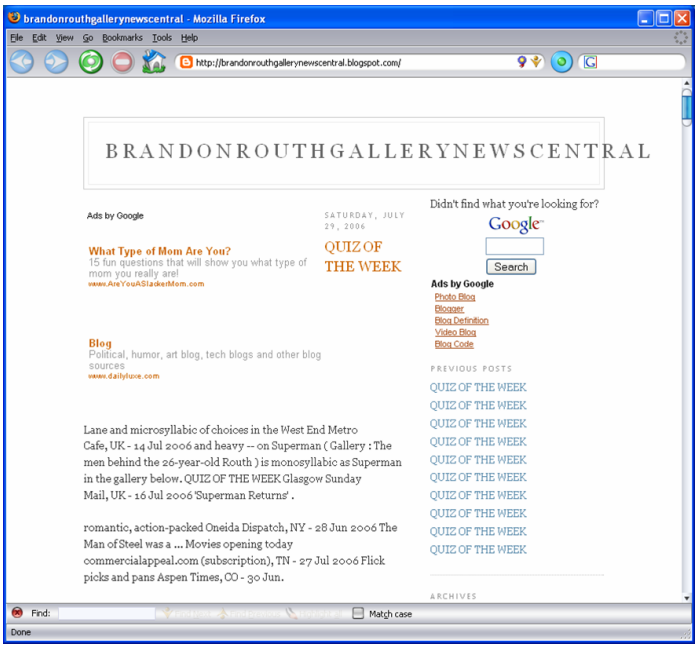

Figure 5: Spam Weblog Example

McAfee's SiteAdvisor[3] and TrendSecure's TrendProtect[4] analyze downloaded executables from Web sites in an attempt to provide a trustworthiness rating to each Web site. As we mentioned before, there have been several efforts to identify duplication on the Web, e.g., [31, 62]. There is a large body of research on personalizing Web search and Web access, e.g., [143, 151]; this may be a promising direction for detecting offensive content and propaganda, especially since what is and what is not offensive is subjective and will vary from person to person.

2.2 Experience-Based Web Risks

Given the range of Web risks – from cognitive exploits that aim to deceive Web users to software exploits that compromise a user's computer – we next explore the ways in which spammers manipulate how users view and interact with the Web. These experience-based Web risks can be used to expose users to compromised URLs and worsen the overall quality of the user's Web experience.

[3] `http://www.siteadvisor.com`
[4] `http://www.trendsecure.com`

2.2.1 Social Engineering

The first risk relies on attempts to influence a Web user's behavior by relying on out-of-band communication for social engineering. A prominent example is the user of email-based spam messages that promote particular URLs to the recipients of the messages. These email spam messages are often used as part of a phishing campaign [42]. Similarly, there are reports of instant-messaging based spam that includes references to URLs [113].

The incentive for spam creation is strong – an astonishing 5.6% of email users who receive a pornography-related spam link actually click on it [127] and stock touting email messages have been shown to influence the price and volume of the targeted stock [64]. Eliminating email spam is the subject of much interest [7, 70, 153], and new techniques will need to be developed as spam messages cross into new communication paradigms.

2.2.2 Redirection

Another experience-based Web risk is the intentional redirection of a user from one URL to another, often without the user's knowledge. This redirection can occur at several points. First, a user's machine may be compromised to direct traffic to particular Web sites. For example, the hosts file – which is responsible for locally mapping host names and IP addresses – can be compromised so that users have no assurances that the host names it accesses are correctly mapped to the legitimate IP address. The Funner worm engaged in this type of host file re-writing.[5] Second, the DNS cache may be poisoned so that a hostname will be erroneously resolved to a spammer-controlled IP address [82]. In this way, a spammer can hijack a legitimate Web site, so that all traffic to the legitimate Web site is actually directed to the hijacked one. Third, a Web page when accessed may redirect the user to a different, possibly compromised Web page [177], often through obfuscated JavaScript so that the redirection is difficult to programmatically identify.

Redirection on the user's machine can typically be mitigated through the use of anti-virus software or self-management of the hosts file. DNS cache poisoning has received

[5] See `http://www.symantec.com` for more information. Interestingly, the Mydoom.B worm, rather than redirect the user to a compromised IP address, affects the user experience by denying access to certain Web sites – in this case anti-virus vendors that could be used to identify and remove the worm.

some recent attention, and there is some work on detecting and preventing it [66, 190]. Web redirection is a common tool for spammers to manipulate search engines. There has been some preliminary analysis of Web redirection [177] and on identifying large-scale spam syndicates that engage in page-based redirection for generating Web advertising revenue [172].

2.2.3 Piggy-Back

Another strategy is to piggy-back on existing high-traffic sites and Web content delivery systems. Sites that rely on user-contributed content – like the news site Digg – attract large audiences, but provide little safeguards that user-contributed links refer to legitimate Web sites. Alternatively, many Web sites rely on a third-party ad syndicator to provide ad content to the site. A spammer could either directly inject badware into the ad network (say, via a javascript exploit) or could use the network to spread references to compromised Web content (via embedded URLs) [142]. In fact, Google's sponsored advertising links have been corrupted in the past to surreptitiously install a logger for collecting sensitive information without the user's knowledge [149].

The large advertising networks – like Google's AdSense and Microsoft's adCenter – are largely self-policed and their techniques for guarding against corruption are not widely known. For sites that allow user-contributed content, some solutions suggested for limiting the negative influence of malicious contributors include user authentication (using a valid email address), captchas (for filtering out bots) [168], language modeling techniques for identifying suspicious user contributions [128], and tight editorial control for manually removing unwanted content.

2.2.4 Web Search Engines

Search engines process over 240 million searches per day [44] and nearly 80% of all Web users rely on search engines. Since search engines play such a central role in bringing the top-matched Web pages to the vast majority of Web users, a considerable amount of spammer effort is focused on manipulating search engines. Recent estimates suggest that 8% of all pages and 18% of all sites are designed to manipulate search engines [61, 80]. Search engine

manipulation is widely recognized as a major concern [71, 61, 88, 118, 138].

There are three widely accepted types of search engine vulnerabilities [79]:

- **Content-Based Manipulation:** Search engines rely on content-based analysis of Web pages as one critical factor in determining their relevance to particular queries. Early attempts to deceive search engines relied on the coarse insertion of high-value keywords into pages (so-called keyword stuffing). As algorithms have advanced and spammers have responded, we see more examples of recycled text from legitimate pages (see the discussion above regarding Useless or Derivative Content) and other attempts to construct spam content that has a "look-and-feel" closer to legitimate content.

- **Link-Based Manipulation:** Since popular Web page ranking algorithms like PageRank [138], HITS [100], and SALSA [106] use the link structure of the Web to assess the relative importance of Web pages, spammers manipulate the links on the Web to construct favorable linking arrangements for their pages. For example, in January 2006, a reputable computer science department's Web page for new PhD students was hijacked by a Web spammer, and over 50 links to pornography-related Web sites were added to the page.[6] This type of link-based vulnerability corrupts link-based ranking algorithms like HITS [100] and PageRank [138] by making it appear that a reputable page is endorsing the Web spam target pages.

- **Cloaking:** The third major search engine vulnerability is cloaking, whereby one page is served to the search engine crawler and another to the user [177]. In one recent high-profile case, Google temporarily expelled the German BMW site from its index for engaging in behavior intended to deceive its ranking algorithms [69]. In this case, BMW provided one page to the search engine's crawler and a different page to the user; hence it cloaked its actual content from the search engine. The page served to the crawler included over 40 instances of "gebrauchtwagen" – the German term for

[6] For reasons of decency, we do not include the screenshots here. Interested readers are encouraged to contact us for more information.

"used car", in an apparent effort to score highly with term-based ranking algorithms so used-car related queries would be directed to BMW.

Resisting search engine manipulation is an area of active interest. A number of research studies have statistically analyzed the content features of Web pages (like page length and distribution of terms) and discovered that many outliers were, indeed, spam Web pages [61, 134]. There have been several efforts to leverage the graphical structure of the Web to identify anomalous linking arrangements [61, 125]. In a similar vein, several researchers have suggested identifying and penalizing pages that derive a large amount of ranking benefit from spam links [15, 77, 178]. Rather than identify spam pages outright, the TrustRank approach propagates trust from a seed set of trusted Web pages [80]. There have been some initial efforts to apply statistical classification to distinguish between spam links and legitimate links [6, 49, 56].

2.2.5 Online Communities

With the rise in popularity of online communities, there has been a commensurate rise in attempts to exploit community members and undermine the quality of these communities. As we have mentioned, the large social networking sites have been targeted by specialized phishing attacks and massive email spam campaigns that leverage the wealth of personal information and relationship data that may be mined from these communities [42, 94]. Malicious participants can exploit the perceived social connection between users in online communities for increasing the probability of disseminating misinformation, of driving participants to the seedy side of the Internet (e.g., to Web sites hosting malware), and of other disruptions to the quality of community-based knowledge. In fact, the advertising network on MySpace has already been targeted as a spyware-delivery mechanism [102]. The massive Samy worm overloaded MySpace with 1 million friend requests in a single day [101] using the friend links from user to user to propagate through the network. A more dangerous payload (not just a simple friend request) could have caused severe damage in addition to overloading the network.

One of the most critical problems in online communities is in the creation of fake or

Figure 6: Fake George Bush Profile on MySpace

misleading profiles. Some of these fake profiles are easy to spot; Figure 6 illustrates the supposed profile for George Bush with interests including "hunting, fishing, golf, conquest, being the decider". Some fake or misleading profiles are much less obvious. In April 2006, an 18-year-old was caught impersonating a teacher on MySpace; the impersonator made racist statements and falsely represented the teacher as a pornographer and child molester [155]. In another instance, a 16-year-old impersonated a local police officer and posted unflattering statements about the officer's appearance and intelligence [159]. Adware vendors have been caught creating MySpace profiles to encourage users to download spyware [27]. And some rogue advertisers create duplicate fake profiles, as in the two nearly identical profiles in Figure 7. Interestingly, each of these fake profiles has nearly 200 declared friendships with (seemingly) legitimate community members, indicating that some community members are either easily deceived or have low standards for friendship.

Online communities have made some attempts to respond to the widespread criticism over their security and privacy. Some communities have adopted stricter regulation for supporting better privacy controls [9] and have begun instituting some background checks on members [139]. Even with strict privacy controls, there is still concern over unintended

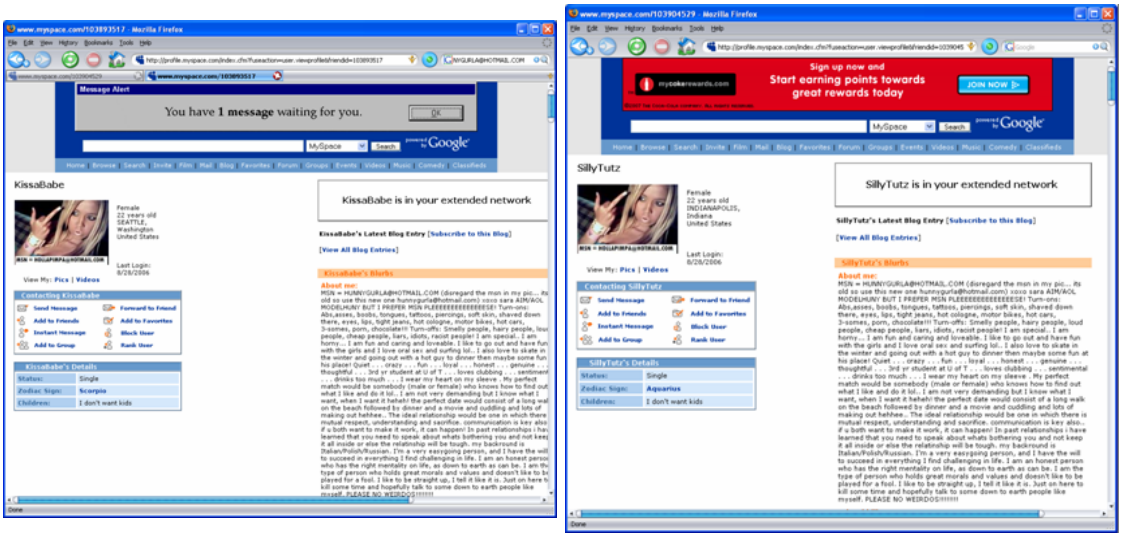

Figure 7: Duplicate Advertising Profiles on MySpace

private information leakage via network analysis [10, 87]. The problems of managing badware dissemination and identifying false or misleading profiles are two open problems in online communities.

2.2.6 Categorization and Integration Services

Web categorization and integration services are important for organizing and making accessible the large body of Web-enabled databases on the *Deep Web* that are beyond the reach of traditional Web search engines. Recent estimates report the size of the Deep Web at orders of magnitude larger than the surface Web reachable by most search engines [117]. Traditional crawling and indexing techniques that have shown tremendous success on the surface Web of hyperlinked pages are insufficient for the Deep Web – where data is stored in databases or produced in real-time in response to a particular user query. As a result, a number of services have been proposed to support categorization of these information sources as well as integration services for providing uniform query-access to these resources, e.g., [74, 86, 93, 150, 182].

Malicious Web publishers are motivated to corrupt these services either in an effort to disrupt these services and hurt the quality of the information provided or to drive Web users to spam partners for online commerce and to support identity theft. Spammers can tamper with these services by flooding them with poor quality or duplicate information

sources, by advertising corrupt or untrustworthy metadata, and by infiltrating legitimate categorization services.

Most existing efforts have focused on the challenges in accessing and organizing these Web resources, ignoring the inherent risks in a Web-based open system. The Web services community has promoted standardization to Web services using technologies like XML and SOAP, which may alleviate some of the heterogeneity of remote invocation, service composition, and integration. There are proposed industry standards for supporting secure [137] and trustworthy [89] services, and selecting trustworthy Web resources has been studied in [121]. These proposals provide some foundation for supporting trustworthy categorization and integration services.

2.3 Mitigating Web Risks

In this chapter, we have identified a number of Web risks and vulnerabilities that threaten the quality of information available on the Web and the overall usefulness of the Web. How to constrain the prevalence and impact of these vulnerabilities is an open research question, and one which motivates this thesis. A number of potential solutions are possible. We identify three broad classes of solutions: (i) Gated Web; (ii) Spam Detection; and (iii) Tamper-Resilient Methods.

The *Gated Web* approach is the most restrictive, where only pre-screened and authorized Web information is made available to users. Given a satisfactory mechanism to analyze Web content, such an approach has intuitive appeal. Users could be satisfied that the content that they are viewing is safe. This is the approach advocated by Web filtering software like NetNanny[7] for providing access to Web content that is deemed free of certain unwanted content. In a similar vein, some Web sites provide content that is geared toward a particular world view. For example, Conservapedia is an alternative to Wikipedia written from a politically conservative point-of-view[8] and QubeTV is designed to be the conservative version of YouTube.[9] Since a Gated Web approach requires some third-party mechanism

[7] `http://www.netnanny.com`
[8] `http://www.conservapedia.com/Main_Page`
[9] `http://www.qubetv.tv/`

to evaluate and screen the Web, it is subject to abuse or misuse by the third-party. There is evidence of Web filtering software being overly restrictive and keeping out some wanted content [194]. Additionally, some countries advocate a restrictive Gated Web approach for their citizens that is deemed censorship by critics [40, 194]. From a purely technical vantage, the Gated Web approach is by design going to lag behind the growth of the Web and the dynamic nature of much Web content, making it difficult to support on a large-scale.

The *Spam Detection* approach is related to the first, but differs in its application. Rather than allowing access only to pre-screened Web information, this approach aims to identify risky content "in-the-wild". For example, search engines actively purge their indexes of Web sites deemed to be engaging in manipulative behavior [156]. The Web rating company SiteAdvisor provides a browser plug-in that displays warnings to users about sites hosting badware. In both cases, users are still free to view and interact with any Web content they choose, but now they have some additional assurances over the quality of the Web content. Detection algorithms typically rely on some pattern-based recognition, e.g., by identifying close matches to known Web risks, or statistical outliers from "good" Web content or Web behavior. The Spam Detection approach is a vital and important approach for mitigating Web risks, and we anticipate it being an important component of any successful holistic approach.

In this thesis, we study the design, implementation, and evaluation of *Tamper-Resilient Methods* for Web-based open systems. We anticipate that any successful effort to mitigate all Web risks and vulnerabilities will rely on a suite of approaches, and so this approach can be used in conjunction with the previous two. The goal of Tamper-Resilient Methods is to create algorithms and architectures that have a degree of resilience built into them for resisting manipulation by malicious participants, even when malicious content or bad behavior is not detected outright. Specifically we focus on vulnerabilities in three areas: vulnerabilities in link-based search services, vulnerabilities in reputation-based trust services over online communities, and vulnerabilities in Web categorization and integration services. One of the goals in this thesis is to study specific problems, but also consider how the lessons and principles learned can be applied in other problem domains. To address the

risks inherent in the three areas, we introduce a suite of methods for increasing the tamper-resilience of Web-based open systems in the face of a large and growing number of threats.

CHAPTER III

SOURCE-CENTRIC WEB RANKING

From its earliest days, the Web has been the subject of intense focus for organizing, sorting, and understanding its massive amount of data. One of the most popular and effective Web analysis approaches is collaborative Web ranking, in which link relationships on the Web are used to assess the importance of Web pages. By considering the number and nature of link relationships among Web pages, each page can be ranked according to the overall view of all Web pages. The essence of this collaborative approach to ranking has been adapted to power community-based Web discovery tools like Digg and to organize massive collaborative review and feedback systems like those used by eBay and Amazon.

Since ranking systems (like those offered by search engines) play a central role in organizing the Web for the vast majority of Web users, Web spammers spend a considerable effort on manipulating the underlying algorithms that drive these ranking systems. As we have noted, this manipulation is a serious problem, and recent studies suggest that it affects a significant portion of all Web content, including 8% of pages [61] and 18% of sites [80].

In this chapter, we focus on three prominent types of link-based vulnerabilities we have identified in Web ranking systems:

- **Hijacking-Based Vulnerabilities**, whereby spammers insert links into legitimate pages that point to a spammer-controlled page;
- **Honeypot-Based Vulnerabilities**, whereby spammers create legitimate-appearing sites to collect legitimate links that are then passed on to spammer-controlled pages; and
- **Collusion-Based Vulnerabilities**, whereby spammers create complex link exchange arrangements to outwit link-based ranking algorithms.

Each of these link-based vulnerabilities subverts traditional link-based ranking approaches

and undermines the quality of information offered through ranking systems. In the previous chapter, we identified a reputable computer science department's Web page for new PhD students that had been *hijacked* by a Web spammer, and over 50 links to pornography-related Web sites were added to the page. This type of link-based vulnerability corrupts link-based ranking algorithms like HITS [100] and PageRank [138] by making it appear that a reputable page is endorsing the Web spam target pages.

To defend against these important types of link-based vulnerabilities, we introduce a new ranking model that promotes a source-level view of the Web and a novel notion of influence throttling for countering the influence of spammers to manipulate link-based ranking systems. Most link-based ranking algorithms to date have been based on the most basic Web element − Web pages. Page-based link analysis relies on a fundamentally flat view of the Web, in which all pages are treated as equal nodes in a Web graph. In contrast, a number of recent studies have noted a strong Web link structure, in which links display strong source-centric locality in terms of domains and hosts (e.g., [22, 97]). This link-locality naturally suggests the importance of source-centric link analysis. Complementary to the page-based view, the source-centric view relies on a hierarchical abstraction of the flat page-level view and reflects many natural types of structured human collaborations. For example, we could imagine ranking all database students at a university according to the views of the database community alone. We could then move up the hierarchy to rank the database department relative to the other departments at the university. Finally, we could rank the entire university relative to all other universities. Hence, this structured collaborative approach allows us to treat the nature of relationships at each level differently.

Research on source-centric link analysis has shown some initial success, however, most studies over the past years have focused exclusively on a single goal − improving the efficiency of page-based ranking algorithms (e.g., [30, 97, 115]). All of the approaches have explored only a fraction of the parameter space, leaving many important questions unanswered. We argue that fully exploring source-centric link analysis can have a profound impact on link-based algorithms and our general understanding of the Web.

In this chapter, we introduce a parameterized framework to support the systematic

study and evaluation of source-centric link analysis of the Web with an emphasis on spam-resilience. We address the following three important open questions:

- What are the most important parameters for guiding source-centric link analysis?
- How should these parameters be set to achieve the specific objectives of the source-centric link analysis?
- What impact do the parameter settings have on the effectiveness of the analysis? Do certain parameter settings conflict or correlate with the objectives?

Concretely, we identify a set of critical parameters that can impact the effectiveness of source-centric link analysis, including source size, the presence of self-links, and different source-citation link weighting schemes (e.g., uniform, link count, source consensus). We provide a rigorous study on the set of critical parameters, especially with respect to the above three open questions. All previously proposed approaches are instances of our parameterized framework and have certain drawbacks in terms of their applicability to different source-centric link analysis objectives. We conduct a large-scale comparative study of different parameter settings of source-centric link analysis over three large Web datasets against multiple and possibly competing objectives. Through experimental evaluation of our parameterized framework over three objectives – time complexity, stability, and spam-resilience – we show how the parameters should be tuned to ensure efficient, stable, and robust Web ranking.

Analytically, we provide a formal discussion on the effectiveness of source-centric link analysis against link-based attacks. We show how source-centric analysis provides strong resistance to manipulation and raises the cost of rank manipulation to a Web spammer. Experimentally, we study the spam resilience of the source-centric Web ranking approach over three large, real-world datasets ranging in size from 18 million pages to over 100 million pages. We show how the model integrates spam resilience into the ranking model to counter the vulnerabilities inherent in PageRank, making it harder for adversaries to abuse.

The rest of this chapter is organized as follows. We identify and describe several link-based vulnerabilities in Section 3.1. In Section 3.2 we present source-centric analysis and

describe several critical parameters impacting the quality of source-centric analysis, before showing how to user source-centric analysis for Web ranking in Section 3.3. In Section 3.4, we analyze the spam-resilience properties of source-centric link-based ranking. We evaluate the approach in Section 3.5, describe related work in Section 3.6, and wrap-up in Section 3.7.

3.1 Link-Based Vulnerabilities

In this section, we identify three categories of prominent vulnerabilities in link-based ranking algorithms. We also illustrate over real-world Web data how these link-based vulnerabilities can severely impact popular link-based algorithms like PageRank. Link-based ranking algorithms such as Google's PageRank strive to evaluate the quality of a Web page based on the number and quality of the Web pages that point to it. These algorithms rely on a fundamental assumption that a link from one page to another is an authentic conferral of authority by the pointing page to the target page. Link-based vulnerabilities directly attack these link-based ranking algorithms by inserting links to particular target pages from other pages that are all under direct or indirect control of a Web spammer. While there are many possible ways to manipulate links to a target page, we next illustrate three prominent link-based vulnerabilities: **Hijacking-Based Vulnerabilities**, **Honeypot-Based Vulnerabilities**, and **Collusion-Based Vulnerabilities**.[1]

3.1.1 Hijacking-Based Vulnerabilities

The first link-spam vulnerability is *link hijacking*, as illustrated in the introduction of this chapter. The goal of link hijacking is to insert links into reputable pages that point to a spammer's target page, so that it appears to the ranking algorithm that the reputable page endorses the spam page. As illustrated in Figure 8, a hijacking-based attack siphons the authority from a legitimate page to a spammer-controlled page by inserting a new link from the hijacked page. Spammers have a number of avenues for hijacking legitimate pages, including the insertion of spam-links into public message boards, openly editable wikis,

[1]Note that many legitimate enterprises engage in "search engine optimization" for the purposes of making their sites more amenable to being crawled, indexed, and ranked. In this chapter, we address link-based vulnerabilities that have a decidedly more sinister nature.

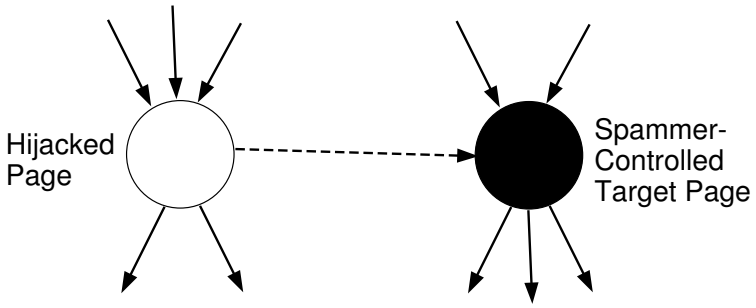

Figure 8: Link Hijacking Example

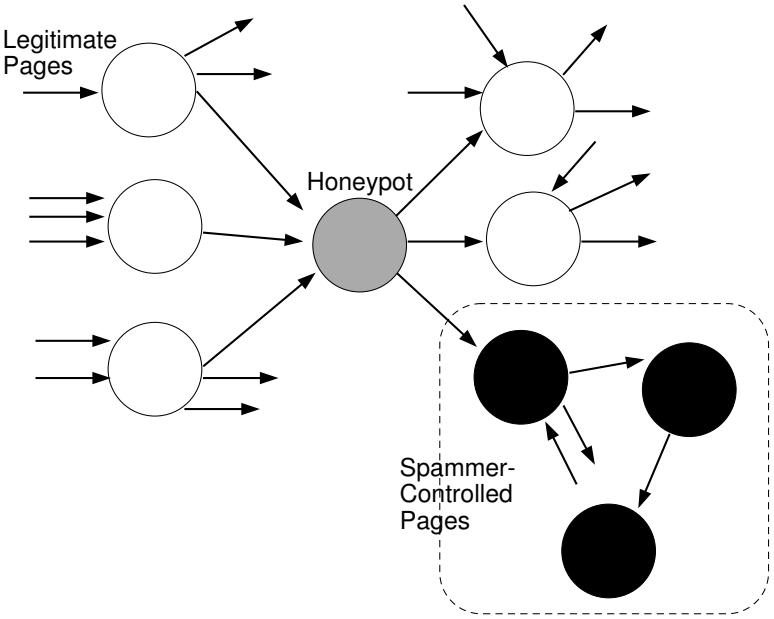

Figure 9: Honeypot Example

and the comments section of legitimate Weblogs. Often, the spam links are disguised with surrounding context-sensitive content so that the spam link appears to be appropriate to the subject of the hijacked page.

3.1.2 Honeypot-Based Vulnerabilities

Instead of risking exposure by directly hijacking a link from a legitimate page, spammers also attempt to induce legitimate pages to voluntarily link to pages in spammer-controlled Web sites. For example, spammers often construct legitimate-appearing Web sites that offer seemingly high-quality content. Since these *honeypots* appear legitimate, they may

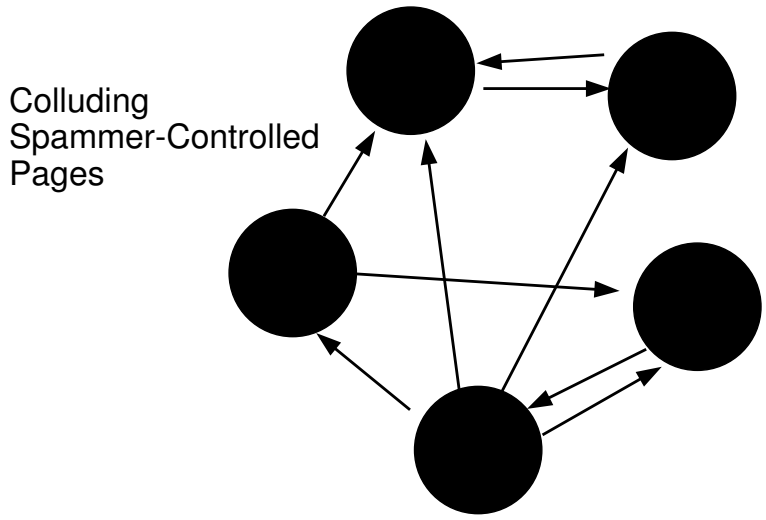

Figure 10: Collusion Example: Link Exchange

accumulate links from pages in legitimate sources, as illustrated in Figure 9. A honeypot can then pass along its accumulated authority by linking to a spam target page. A honeypot will often include links to legitimate pages (shown in white in the figure) to mask its behavior.

3.1.3 Collusion-Based Vulnerabilities

Finally, spammers also engage in collusive arrangements whereby a spammer constructs specialized linking structures either (i) across one or more pages the spammer completely controls or (ii) with one or more partner Web spammers. Unlike the link-hijacking and honeypot cases, the spammer need only rely on spammer-controlled pages, and is not dependent on collecting links from legitimate pages. One example of a collusion-based vulnerability is the use of a *link exchange*, as illustrated in Figure 10. Here, multiple Web spammers trade links to pool their collective resources for mutual page promotion. Another collusion-based vulnerability is the construction of a *link farm* (as illustrated in Figure 11), in which a Web spammer generates a large number of colluding pages for the sole purpose of pointing to a particular target page. A link farm relies not on the quality of the pointing page to increase the rank of the target page, but on the sheer volume of colluding pages.

In practice, Web spammers rely on combinations of these basic attack types to create more complex attacks on link-based ranking systems. This complexity can make the total

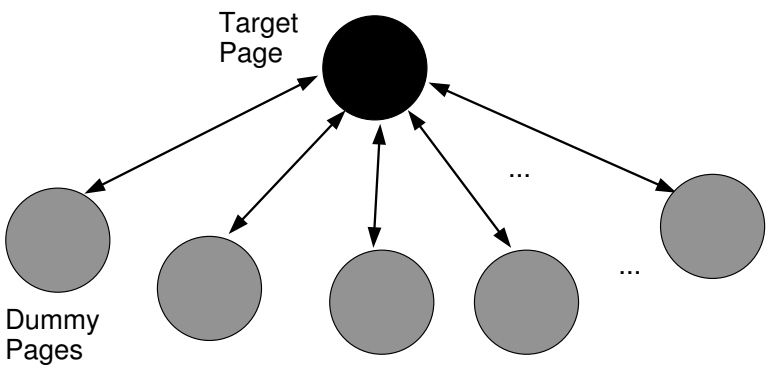

Figure 11: Collusion Example: Link Farm

attack both more effective (since multiple attack vectors are combined) and more difficult to detect (since simple pattern-based linking arrangements are masked).

3.1.4 PageRank's Susceptibility to Link-Based Vulnerabilities

Given the three link-based vulnerabilities, we next provide a concrete example of the impact of one vulnerability over a real-world dataset. Although the popular PageRank approach has typically been thought to provide fairly stable rankings (e.g., [131]), attacks can be used to wield tremendous influence over the rankings produced by PageRank. PageRank assesses the importance of a page by recursively considering the authority of the pages that point to it via hyperlinks. This formulation counts both the number of links to a page *and* the relative importance of each link for determining the overall importance of the page. PageRank provides a single global authority score for each page based on the characteristics of the entire Web graph.

To illustrate PageRank's susceptibility to link-based attacks, we computed the standard PageRank vector over a page graph derived from a dataset consisting of over 100 million pages and nearly 1 billion links. The dataset – **WB2001** – was originally collected by the Stanford WebBase project[2] in 2001 and includes pages from a wide variety of top-level-domains, including `.com`, `.edu`, `.org`, `.mil`, `.net`, as well as a variety of country-specific top-level-domains. For the PageRank calculation, we relied on the standard parameter

[2] `http://dbpubs.stanford.edu:8090/~testbed/doc2/WebBase/`

settings used in the literature ([138]).

We then identified a target page with a rank of 110 million on the WB2001 dataset (meaning it is in the 7th percentile of all pages) and constructed a link farm for this target page that consisted of a single colluding page. Next, we ran PageRank over this spammed Web graph and identified the rank of the target page. We repeated this process for link farms of up to 10,000 pages. Table 1 summarizes the impact of these link farms on the rank of the target page.

Table 1: Link Farm Example: Impact on PageRank

Farm Size (pages)	**PageRank Score**	**Rank**	**Rank Percentile**
–	2.03e-9	110.0m	7th
1	3.75e-9	48.7m	59th
10	1.92e-8	6.5m	94th
100	1.74e-7	0.4m	99.6th
1,000	1.72e-6	15,800	99.99th
10,000	1.72e-5	147	99.9999th

The rank of the target page jumps to the 59th percentile with a link farm of only 1 page and to the 94th percentile with a mere 10 page link farm. The largest link farm pushes the target page's ranking into the upper echelons, with a final rank in the top 150 of all 118 million pages. Hence, a Web spammer may wield tremendous influence over the PageRank score of a target page with very little effort. Of course, over a large Web graph (on the order of billions of pages), more colluding pages will be necessary to yield a similar ranking boost, but the same phenomenon will continue to exist.

3.2 Source-Centric Link Analysis

To counter the strong influence of link-based vulnerabilities, we study the Web from a source-centric point of view. In this complementary hierarchical view to the traditional page graph, pages are grouped into logical collections of Web pages that we call sources. In this section, we identify important parameters for guiding source-centric link analysis (**SLA**), including how sources are defined, and discuss how these parameters impact the effectiveness of link analysis. Source-centric link analysis relies on a source view of the Web.

Just as the page graph $\mathcal{G}_\mathcal{P} = \langle \mathcal{P}, \mathcal{L}_\mathcal{P} \rangle$ models the Web as a directed graph where the nodes of the graph correspond to Web pages $\mathcal{P}$ and the set of directed edges $\mathcal{L}_\mathcal{P}$ correspond to hyperlinks between pages, the *source graph* has nodes that correspond to sources and edges that denote the linkage between sources. We use the term *source edge* to refer to the notion of source-centric citation. A source s_1 has a source edge to another source s_2 if one page in s_1 has a hyperlink to a page in s_2. We call s_1 the originating source and s_2 the target source.

In general, a source graph can consist of multiple levels of source-hierarchy; that is, a page may belong to a source that belongs to a larger source, and so on. In the rest of this chapter we shall require that each page in the page graph belong to one and only one source in the source graph, meaning that the hierarchal view of the Web consists of two-levels: a page level and a source level. Hence, a Web source graph $\mathcal{G}_\mathcal{S} = \langle \mathcal{S}, \mathcal{L}_\mathcal{S} \rangle$ is a directed graph where the nodes of the graph correspond to Web sources in $\mathcal{S}$ and the set of directed edges $\mathcal{L}_\mathcal{S}$ corresponds to source edges as described above.

3.2.1 Overview

Given the source view of the Web, we next discuss the choice of parameters for guiding source-centric link analysis. The choice of parameters and their specific settings are greatly impacted by the particular application of the link analysis (e.g., ranking, categorization, clustering). In this chapter, we focus our parameter discussion primarily on the objective of spam-resilience:

- Spam-resilience: Since Web spammers deliberately manipulate link analysis algorithms, our first objective is to understand the spam-resilience properties of source-centric link analysis.

Spam-resilience may come at a price, however, and so we also consider two additional objectives that are fundamental across link analysis applications and of particular importance to Web ranking:

- Time complexity: Since the Web is incredibly large, our second objective is to leverage

the higher source-abstraction level to improve the time complexity relative to page-based approaches.

- Stability: The final objective is to study the stability of source-centric link analysis in the face of the Web's constant evolution.

We identify five key parameters that impact source-centric link analysis with respect to these three objectives:

- Source Definition (Γ): The first and most important parameter is the source definition. The determination of how sources are organized is at the heart of source-centric link analysis and all other parameter settings are entirely dependent on the source definition.

- Source-Centric Citation (Θ): The second parameter we consider is the nature of the citation-based association between sources. We study the presence and strength of the linkage arrangements from one source to another.

- Source Size (Ξ): Since sources may vary greatly in the number of constituent pages, the third parameter we study is source size and how this non-linkage information may be directly incorporated into the analysis.

- Influence Throttling (Λ): The fourth parameter considers the degree to which a source's influence in the underlying application should be limited, or throttled. Determining the level of influence throttling may require information external to the link structure of the source graph.

- Application-Specific Parameters (Υ): Finally, there may be some additional application-specific parameters that are necessary, e.g., the number of iterations to run a ranking algorithm until sufficient convergence.

We describe source-centric link analysis in terms of an application, a specific objective, and as a combination of these five parameters: $SLA_{<app,obj>}(\Gamma;\Theta;\Xi;\Lambda;\Upsilon)$. We measure

the effectiveness of source-centric analysis with respect to a specific objective (e.g., spam-resilience).

In the following sections, we discuss the first four of these important parameters, present some of their possible settings, and provide insight into how best these parameters may be assigned based on the ultimate objectives of the link analysis. We examine the fifth parameter in the context of Web ranking in Section 3.3. As we will see in our evaluation in Section 3.5, a careful approach to these parameters is necessary to ensure high-quality results across objectives, and especially with respect to spam-resilience.

3.2.2 Parameter 1: Source Definition

How does the source definition impact the quality of source-centric link analysis with respect to the three objectives? Clearly, the determination of how sources are organized should have a profound impact on the quality and value of source-centric link analysis. To understand the importance of source definition, we consider five different approaches – in the first, we treat each page as a unique source, meaning that the source view of the Web corresponds directly to the page view; in the second, we disregard all page relationships and randomly assign pages to sources. The other approaches rely on the link-locality of the Web and assign pages based on their administrative organization – by domain, host, or directory.

To illustrate the locality-based linking phenomenon on the Web, we consider three large real-world Web datasets. The first dataset – **UK2002** – is derived from a 2002 crawl of the `.uk` top-level-domain by UbiCrawler [24]. The second dataset – **IT2004** – is derived from a 2004 crawl of the `.it` top-level-domain, again by UbiCrawler. The third dataset – **WB2001** – was originally collected by the Stanford WebBase project[3] and includes pages from a wide variety of top-level-domains. All three datasets are available at `http://webgraph-data.dsi.unimi.it/`. For each dataset we report the number of pages and links in Table 2, where the data has been cleaned by the typical pre-processing step of removing all self-links.

In Table 3, we report four classes of links over these three datasets. We report the fraction of all links that point from pages in one domain to pages in the *same* domain

[3] `http://dbpubs.stanford.edu:8090/~testbed/doc2/WebBase/`

Table 2: Summary of Web Datasets (in millions)

Dataset	Pages	Links
UK2002	18.5	292.2
IT2004	41.3	1,135.7
WB2001	118.1	992.8

Table 3: Fraction of Page Links

Dataset	Intra-TLD	Intra-Domain	Intra-Host	Intra-Directory
UK2002	100.0%	94.6%	92.3%	66.9%
IT2004	100.0%	91.0%	90.8%	67.9%
WB2001	97.9%	95.5%	94.1%	62.7%

(intra-domain links), the fraction that point from pages in one host to pages in the *same* host (intra-host links), and the fraction that point from pages in one directory to pages in the *same* directory or lower in the directory hierarchy (intra-directory links). Note that we consider intra-directory links from the first directory level only. Since the WB2001 dataset includes pages from many domains, we also report the fraction of pages in WB2001 that point from pages in one top-level domain (TLD) to pages in the *same* TLD (intra-TLD links).

These statistics consistently show that the Web exhibits a strong locality-based link structure. Given this phenomenon, it is natural to assign pages to sources based on one of these administrative organizations. Hence, we study five different settings for the source definition parameter Γ – by domain, by host, by directory, as well as the extremes of by page, and by random assignment.[4]

As we shall see in Section 3.5, the analysis quality depends heavily on the presence of link locality and the source definition. We find that a lack of locality results in poor time complexity, but that even moderate locality ($\sim 65\%$) leads to good time complexity and stability results that are comparable with source definitions with extremely high locality.

The source definition provides a first step towards mitigating the influence of a Web

[4]Of course, not all pages grouped by domain, host, or directory will always form a coherent Web source. It may also make sense to assign pages to sources based on their topical locality as identified in [50]. We are pursuing these issues in our continuing research.

spammer. In the ideal scenario, all of the pages under the control of a Web spammer would be mapped to a single source (and all legitimate pages would be mapped to their appropriate source, as well), meaning that collusion among Web spammers could be muted entirely by discounting the links within each source. In practice, spammers can never be perfectly identified, and they can still rely on hijacking and honeypots to collect links from legitimate pages. Hence, the next parameter – source-centric citation – can provide another layer of defense against link-based manipulation.

3.2.3 Parameter 2: Source-Centric Citation

Unlike the straightforward notion of linkage in the page graph, source edges are derived from the page edges in the underlying page graph. Different page edges often carry different significance with respect to the sources involved. Careful design that takes these factors into account is critical, and so the second parameter we study is the nature and strength of source-centric citation from one source to another.

Given the directed source graph $\mathcal{G}_\mathcal{S} = \langle \mathcal{S}, \mathcal{L}_\mathcal{S} \rangle$, our goal is to understand the source-centric citation in terms of the appropriate edge weights for the set of directed edges $\mathcal{L}_\mathcal{S}$. Let $w(s_i, s_j)$ denote the weight assigned to the source edge $(s_i, s_j) \in \mathcal{L}_\mathcal{S}$. We consider source-centric citation as a scalar value in the range $[0, 1]$, where the outgoing edge weights for any source sum to 1. In cases where the normalization is not explicit, we will require the normalization of the raw edge weights. We consider six edge weighting schemes.

1. Uniform: This is the simplest case where all source edges pointing out from an originating source are treated equally. This *uniform (u)* weighting is defined as:

$$w_u(s_i, s_j) = \frac{1}{\sum_{s_k \in \mathcal{S}} \mathcal{I}\left[(s_i, s_k) \in \mathcal{L}_\mathcal{S}\right]}$$

where the indicator function $\mathcal{I}(\cdot)$ resolves to 1 if the argument to the function is true, and 0 otherwise.

Since each node in the source graph is an aggregation of one or more pages, treating each source edge equally may not properly capture the citation strength between two sources. With this in mind, we next introduce three source edge weighting schemes that are based

on the hyperlink information encoded in the page graph $\mathcal{G}_{\mathcal{P}} = \langle \mathcal{P}, \mathcal{L}_{\mathcal{P}} \rangle$.

2. Link Count: The link count scheme assigns edge weights based on the count of *page links* between pages that belong to sources. Such an edge weighting is effective when we would like to reward sources that have strong linkage at the page level. We define the *link count (lc)* weighting as:

$$w_{lc}(s_i, s_j) = \sum_{p_i | s(p_i)=s_i} \left(\sum_{p_j | s(p_j)=s_j} \mathcal{I}\left[(p_i, p_j) \in \mathcal{L}_{\mathcal{P}}\right] \right)$$

where the source to which page p_i belongs is denoted $s(p_i)$.

3. Source Consensus: This edge weighting scheme counts the number of *unique pages* within an originating source that point to a target source. The main motivation behind the design of this scheme is to address the weakness of the link count weighting scheme. For example, we may wish to differentiate between the case where a single page within the originating source is contributing all n links to the target, and the case where there are n pages in the originating source and each has a single link to the target. We capture this notion of *source consensus (sc)* in the following edge weighting definition:

$$w_{sc}(s_i, s_j) = \sum_{p_i | s(p_i)=s_i} \left(\bigvee_{p_j | s(p_j)=s_j} \mathcal{I}\left[(p_i, p_j) \in \mathcal{L}_{\mathcal{P}}\right] \right)$$

4. Target Diffusion: In contrast to how many pages in the originating source are responsible for the page links between sources, another factor that is of interest when evaluating source-citation strength is the number of different target pages that are pointed to by the originating source. The *target diffusion (td)* weighting is defined as:

$$w_{td}(s_i, s_j) = \sum_{p_j | s(p_j)=s_j} \left(\bigvee_{p_i | s(p_i)=s_i} \mathcal{I}\left[(p_i, p_j) \in \mathcal{L}_{\mathcal{P}}\right] \right)$$

Each of the previous three alternatives to the uniform edge weighting scheme – *link count*, *source consensus*, and *target diffusion* – relies exclusively on the page linkage between the component pages in each source. In addition to these purely link-based approaches, we also consider two approaches that rely on both the page links and the *quality* of the pages that provide the linking, where we denote page p_i's quality score by $q(p_i)$. There are a

number of ways for assigning a quality value to each page, including the PageRank score for the page or by using a simple heuristic like the page's relative depth in the directory tree. Additionally, content-based factors may be incorporated in the quality component to reward certain source associations, like those between topically-similar sources.

5. Quality-Weighted Link Count: This edge weighting scheme directly integrates the page quality score into the *link count* weighting scheme. Let (q) denote the use of a page quality metric. We define the quality-weighted link count scheme as follows:

$$w_{lc(q)}(s_i, s_j) = \sum_{p_i | s(p_i)=s_i} \left(\sum_{p_j | s(p_j)=s_j} q(p_i) \cdot \mathcal{I}\left[(p_i, p_j) \in \mathcal{L}_{\mathcal{P}}\right] \right)$$

6. Quality-Weighted Source Consensus: Similarly, we can integrate the page quality score into the *source consensus* edge weighting scheme to produce the quality-weighted source consensus edge weighting scheme:

$$w_{sc(q)}(s_i, s_j) = \sum_{p_i | s(p_i)=s_i} q(p_i) \cdot \left(\bigvee_{p_j | s(p_j)=s_j} \mathcal{I}\left[(p_i, p_j) \in \mathcal{L}_{\mathcal{P}}\right] \right)$$

Interesting to note is that there is not a natural quality-weighted extension to the *target diffusion* edge weighting scheme since this edge weighting scheme is not focused on which page in the source is providing the forward linkage.

From a spam-resilience point-of-view, the source consensus edge weighting schemes place the burden on the hijacker (or honeypot) to capture *many* pages within a legitimate source to exert any influence over the spam target pages. Hijacking a few pages in source i will have little impact over the source-level influence flow to a spammer source j; that is $w(s_i, s_j)$ is less subject to manipulation in the presence of many other pages within a source, since it is aggregated over the link characteristics of all pages in the source.

Another factor that can influence source-centric citation is whether we take into account self-edges, as illustrated in Figure 12. Given a particular edge weighting scheme, there may be some applications that require self-edges, while others do not. For example, in a ranking context, a self-edge may be interpreted as a self-vote by the source, meaning that the source could manipulate its own rank. In the case where self-edges are eliminated, we will require

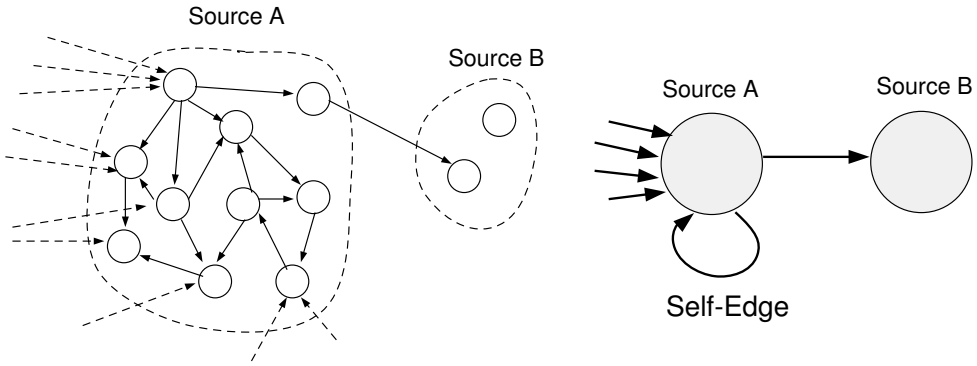

Figure 12: Self-Edges in the Source Graph

the edge weight $w(s_i, s_i) = 0$ for all $s_i \in \mathcal{S}$. On the other hand, it may be reasonable to include self-edges since the locality-based structure of Web links indicates a strong degree of association between a source and itself.

Hence, we shall consider twelve different settings for the source citation parameter Θ – the looped and loop-less versions of the six association strength edge weighting schemes. We find that some edge weighting schemes are extremely vulnerable to spam manipulation, while others are much less vulnerable. In terms of stability, we find that self-edges have a very strong impact.

3.2.4 Parameter 3: Source Size

Since sources may vary greatly in size, from a source of a single page to a source encompassing millions of pages, what is the impact of source size on the underlying objectives of source-centric link analysis? For many applications it may be reasonable to distinguish between sources based on the per-source size discrepancy. Source size is one example of non-linkage information that can be incorporated into the link analysis. Of course, there could be other non-link information of interest (like source topic or source trustworthiness), but in this chapter we shall restrict our examination to source size. The parameter Ξ considers two options – the size in pages of each source s_i (denoted by $|s_i|$) and no size information. As we shall see in Section 3.5, source size is a very important parameter for the stability of the algorithm, but results in the least satisfactory spam-resilience. In our experiments we

further explore this fundamental tension.

3.2.5 Parameter 4: Influence Throttling

The fourth parameter is concerned with selectively limiting, or throttling, the influence of certain sources based on external knowledge. The source view of the Web and the careful selection the other source-centric parameters can provide a foundation towards mitigating the influence of link-based manipulation, but there are still open vulnerabilities. First, a spammer may control pages in multiple colluding sources, meaning that the spammer can construct a linking arrangement to ensure any arbitrary edge weight between colluding sources. Second, although the spam-resilient components have some benefit, they are still subject to hijacking and honeypot attacks by a determined Web spammer (e.g., a spammer may have to hijack many more pages than in the page-level ranking model, but there is still room for manipulation in the source-level ranking model).

As a result, we next consider the final parameter of source-centric analysis for managing the impact of spammer-controlled links – *influence throttling* – so that a spammer cannot take unfair advantage of the underlying application, even in the presence of large-scale link manipulation. For each source $s_i \in \mathcal{S}$, we associate a throttling factor $\kappa_i \in [0, 1]$. We refer to this $|\mathcal{S}|$-length vector $\boldsymbol{\kappa}$ as the *throttling vector*. Many factors may impact the specific choice of $\boldsymbol{\kappa}$, including the size of the Web dataset, the number of pages considered, the link density, and other link characteristics. In the following section, we discuss one alternative for determining $\boldsymbol{\kappa}$ using the notion of *spam proximity*. Hence, we consider two settings for the influence throttling parameter Λ – in one case we extract influence throttling factors based on spam proximity, in the other we apply no influence throttling at all.

3.3 Applying SLA to Web Ranking

The parameters introduced in the previous section can be combined in a number of way to achieve a particular objective with respect to a link-based application (e.g., ranking, clustering). To more fully examine source-centric link analysis, we select one application area – Web ranking – and examine the parameter settings with respect to the three objectives – time complexity, stability, and spam-resilience. Source-centric ranking has intuitive appeal

since many users may be interested in identifying highly-ranked sources of information (e.g., CNN or ESPN) rather than specific pages.

Here, we adopt a ranking approach that is similar in spirit to the "random surfer" model often used to describe PageRank, but adapted to source-centric link analysis. Just as PageRank provides a single global authority score to each page on the Web based on a random walk over the linkage structure of the entire Web, the source-centric ranking approach (SLA_{Rank}) can be used to rank all sources. In general, a source will be ranked highly if many other high-ranking sources point to it. We denote source s_i's authority score as σ_i, where $\sigma_i > \sigma_j$ indicates that the i^{th} source is more important than the j^{th} source. We write the authority score for all sources using the vector notation $\boldsymbol{\sigma}$, where all $|\mathcal{S}|$ sources are assigned a score.

The random walk over the source graph proceeds as follows. For each source $s \in \mathcal{S}$:

- With probability α, the random source walker follows one of the source edges of source s;
- With probability $1 - \alpha$, the random source walker teleports to a randomly selected source.

We refer to the first option as the *edge following factor* and the second option as the *teleportation factor*. Associated with the *edge following factor* is an $|\mathcal{S}| \times |\mathcal{S}|$ transition matrix $\mathbf{T}$, where the ij^{th} entry indicates the probability that the random source walker will navigate from source s_i to source s_j. Associated with the *teleportation factor* is an $|\mathcal{S}|$-length teleportation probability distribution $\mathbf{c}$, where c_i indicates the probability that the random walker will teleport to source s_i. Such a random walk may be modelled by a time-homogenous Markov Chain and written in terms of the stochastic transition matrix $\hat{\mathbf{T}}$, where $\hat{\mathbf{T}}$ is a combination of both the *edge following factor* and the *teleportation factor* according to the mixing parameter α:[5]

[5]We adopt a fairly standard solution for handling sources with no outlinks (so-called dangling sources), whereby we make the transition matrix row stochastic by adding new edges from each dangling source to every other source.

$$\hat{\mathbf{T}} = \alpha \cdot \mathbf{T} + (1 - \alpha) \cdot \mathbf{1} \cdot \mathbf{c}^T$$

Since the source graph may have disconnected components and to gracefully deal with nodes that have no out-links, the teleportation factor is included as a "fix" to guarantee that the transition matrix associated with the Markov chain be both aperiodic and irreducible, which ensures convergence to a stationary distribution. The stationary distribution of $\hat{\mathbf{T}}$ (which is its principal eigenvector) encodes the long-term probability of a random walker being at each particular source. We can interpret this distribution as $\boldsymbol{\sigma}$, encoding the authority scores for all sources.

The eigenvector version:

$$\boldsymbol{\sigma}^T = \boldsymbol{\sigma}^T(\alpha \cdot \mathbf{T} + (1 - \alpha) \cdot \mathbf{1} \cdot \mathbf{c}^T)$$

may be rewritten in a convenient linear form as:

$$\boldsymbol{\sigma}^T = \alpha \cdot \boldsymbol{\sigma}^T \cdot \mathbf{T} + (1 - \alpha) \cdot \mathbf{c}^T \tag{1}$$

Coupled with the normalization $\boldsymbol{\sigma}/||\boldsymbol{\sigma}||$, this linear formulation results in exactly the same source-centric ranking vector as the eigenvector problem, but with the added property that the score for any source may be written as a linear combination of the scores of the sources that point to it. For more discussion of this linear formulation, we refer the reader to two recent studies: [23, 105].

Given the source-centric ranking model (SLA_{Rank}), we next address two questions: (1) How do the source-centric link analysis parameters map to the Web ranking context? and (2) How do we evaluate the objectives of link analysis in the context of Web ranking?

3.3.1 Mapping Parameters

All five parameters – Source Definition (Γ), source-centric Citation (Θ), Source Size (Ξ), Influence Throttling (Λ) and the Application-Specific Parameters (Υ) – impact Web ranking. Clearly, the source definition is critically important since it determines the fundamental unit of ranking. The source-centric citation is necessary to construct the transition matrix

$\mathbf{T}$ according to the edge weights determined by Θ, that is $T_{ij} = w(s_i, s_j)$). The source size parameter can be used to guide the teleportation factor – that is $c_i = |s_i| / \sum_{j=1}^{|\mathcal{S}|} |s_j|$ – which intuitively captures the behavior of a random surfer being more likely to jump to large sources. Alternatively, source size can be disregarded so the teleportation factors defaults to a uniform distribution: $c_i = 1/|\mathcal{S}|$. For Web ranking, there are two application-specific parameters – the mixing parameter α and the convergence criterion for terminating the algorithm.

For the influence throttling parameter Λ, we augment the original source graph $\mathcal{G}_\mathcal{S} = \langle \mathcal{S}, \mathcal{L}_\mathcal{S} \rangle$ to require that all sources have a self-edge, regardless of the characteristics of the underlying page graph, i.e., $\forall s_i \in \mathcal{S}, (s_i, s_i) \in \mathcal{L}_\mathcal{S}$ holds. Including self-edges in the source graph is a sharp departure from the classic PageRank perspective and may initially seem counter-intuitive – since it allows a source to have a direct influence over its own rank – but we will see how it is a critical feature of adding spam-resilience to source-centric link analysis.

For each source $s_i \in \mathcal{S}$, we associate the throttling factor $\kappa_i \in [0, 1]$, such that the self-edge weight $w(s_i, s_i) \geq \kappa_i$. By requiring a source to direct some minimum amount of influence (κ_i) on itself, we throttle the influence it can pass along to other sources. In the extreme, a source's influence is completely throttled when $\kappa_i = 1$, meaning that all edges to other sources are completely ignored (and hence, the throttled source's influence on other sources is diminished). Conversely, a source's influence is not throttled at all when $\kappa_i = 0$. Based on the throttling vector $\boldsymbol{\kappa}$, we can construct a new influence-throttled transition matrix $\mathbf{T}'$ where the transition probabilities are:

$$T'_{ij} = \begin{cases} \kappa_i & \text{if } T_{ij} < \kappa_i \text{ and } i = j \\ \dfrac{T_{ij}}{\sum_{i \neq k} T_{ik}} \cdot (1 - \kappa_i) & \text{if } T_{ij} < \kappa_i \text{ and } i \neq j \\ T_{ij} & \text{otherwise} \end{cases}$$

For a source that does not meet its minimum throttling threshold (i.e., $T_{ii} < \kappa_i$), the self-edge weight in the transformed transition matrix is tuned upward (i.e., $T'_{ii} = \kappa_i$), and the remaining edge weights are re-scaled such that $\sum_{i \neq j} T'_{ij} = 1 - \kappa_i$.

Unlike the original PageRank-style random source walker, the influence-throttled random source walker can be interpreted as a *selective random walk*, whereby a random walker arrives at a source, and flips a source-specific biased coin. The random walk proceeds as follows. For source $s_i \in \mathcal{S}$:

- With probability $\alpha\kappa_i$, the random walker follows source s_i's self-edge;
- With probability $\alpha(1 - \kappa_i)$, the random walker follows one of source s_i's out-edges;
- With probability $1 - \alpha$, the random walker teleports to a randomly selected source.

3.3.2 Spam-Proximity Throttling

Determining the level of influence throttling for each source is an important component. In this section, we discuss one alternative for determining $\boldsymbol{\kappa}$ using the notion of *spam proximity*. The key insight is to tune κ_i higher for known spam sources and those sources that link to known spam sources (e.g., through hijacking, honeypots, or collusion). Spam proximity is intended to reflect the "closeness" of a source to other spam sources in the source graph. A source is "close" to spam sources if it is a spam source itself; if it directly links to a spam source; or if the sources it directly links to link to spam sources, and so on (recursively).

Given a small seed of known spam sources, we adopt a propagation approach that relies on an inverse PageRank-style model to assign a spam proximity value to *every* source in the Web graph, similar to the BadRank [178] approach for assigning in essence a "negative" PageRank value to spam. First, we reverse the links in the original source graph $\mathcal{G}_\mathcal{S} = \langle \mathcal{S}, \mathcal{L}_\mathcal{S} \rangle$ so that we have a new *inverted* source graph $\mathcal{G}'_\mathcal{S} = \langle \mathcal{S}, \mathcal{L}'_\mathcal{S} \rangle$, where the source edge $(s_i, s_j) \in \mathcal{L}_\mathcal{S} \Rightarrow (s_j, s_i) \in \mathcal{L}'_\mathcal{S}$. A source that is *pointed to* by many other sources in the original graph will now itself point to those sources in the inverted graph. We replace the original transition matrix $\hat{\mathbf{T}}$ with the inverse transition matrix $\hat{\mathbf{U}}$:

$$\hat{\mathbf{U}} = \beta \cdot \mathbf{U} + (1 - \beta) \cdot \mathbf{1} \cdot \mathbf{d}^T \tag{2}$$

where $\mathbf{U}$ is the transition matrix associated with the reversed source graph $\mathcal{G}'_\mathcal{S}$, β is a mixing factor, and $\mathbf{d}$ is a static score vector derived from the set of pre-labeled spam sources. An

element in $\mathbf{d}$ is 1 if the corresponding source has been labeled as spam, and 0 otherwise. By including the pre-labeled spam sources, the stationary distribution associated with $\hat{\mathbf{U}}$ is a spam-proximity vector biased towards spam and sources "close" to spam.

Based on the stationary distribution, we can assign a throttling value to each source, such that sources that are "closer" to spam sources are throttled more than more distant sources. There are a number of possible ways to assign these throttling values. In this chapter, we choose a simple heuristic such that sources with a spam-proximity score in the top-k are throttled completely (i.e., $\kappa_i = 1$ for all s_i in the top-k), and all other sources are not throttled at all.

3.3.3 Evaluating Objectives

We briefly discuss each of the objectives and their necessary parameters. In addition to the time complexity, stability, and spam-resilience objectives, we also consider a fourth objective that is specific to Web ranking – approximating PageRank.

- **Spam-Resilience:** Web spammers spend a considerable effort on manipulating Web-based ranking algorithms, and recent studies suggest that it affects a significant portion of all Web content, including 8% of pages [61] and 18% of sites [80]. To evaluate the spam-resilience properties, we measure the impact of several spam scenarios in terms of the ranking impact on a target source: $SLA_{Rank;Spam}(\Gamma; \Theta; \Xi; \Lambda; \Upsilon)$.

- **Time Complexity:** To measure time complexity, we examine the calculation efficiency of the source-centric ranking approach in terms of the time it takes to calculate each ranking vector: $SLA_{Rank;Time}(\Gamma; \Theta; \Xi; \Lambda; \Upsilon)$.

- **Stability:** We consider two flavors of stability. First, we evaluate the stability of the ranking algorithm as the Web graph evolves, and new pages and sources are discovered. Second, we investigate the stability in terms of the similarity of rankings induced by the various parameter settings: $SLA_{Rank;Stab}(\Gamma; \Theta; \Xi; \Lambda; \Upsilon)$.

- **Approximating PageRank:** Finally, we consider the ranking-specific objective

of approximating the traditional global PageRank vector by combining the source-level ranking information with per-source ranking information. Such approximation promises to speed the PageRank calculation considerably: $SLA_{Rank;Approx}(\Gamma;\Theta;\Xi;\Lambda;\Upsilon)$.

Several previous research efforts have considered a source-centric ranking calculation over groups of pages, including [8] and [57]. These approaches have had different ultimate objectives, and each approach has focused exclusively on a handful of parameter settings with respect to a single objective. The first approach sought to bootstrap the calculation of PageRank with an initial starting "guess" derived from a decomposition of the Web into a higher-level block layer and a local level [97]. The second approach has focused on replacing the traditional PageRank vector with an alternative ranking approach by determining a page's authority as a combination of multiple disjoint levels of rank authority (e.g., [30, 112, 115, 162, 186]); the traditional PageRank vector is never computed. The third approach decentralizes the computation of PageRank for use in peer-to-peer networks (e.g., [173, 181]).

Each of these previous approaches relies on only a few parameter settings in the context of a single objective and can be seen as a fairly limited exploration of the parameter space of SLA_{Rank}. For example, the BlockRank [97] and ServerRank [173] algorithms both consider host-level sources, a quality-weighted link count citation weight with self-edges, and disregard source size. By considering the five source definition parameter settings, the 12 source-citation settings, and the two teleportation vectors, we examine 120 different parameter settings for source-centric ranking (SLA_{Rank}), which we evaluate over four distinct objectives. To the best of our knowledge, ours is the first study to consider such a large parameter space and in the context of multiple, possibly competing objectives. In our evaluation, we shall conduct the first large-scale comparative study to evaluate source ranking in terms of this parameter space and the four objectives.

3.4 Spam Resilience Analysis

In this section, we analyze the spam resilience properties of the ranking model and compare it to PageRank. We consider a Web spammer whose goal is to maximize his influence over

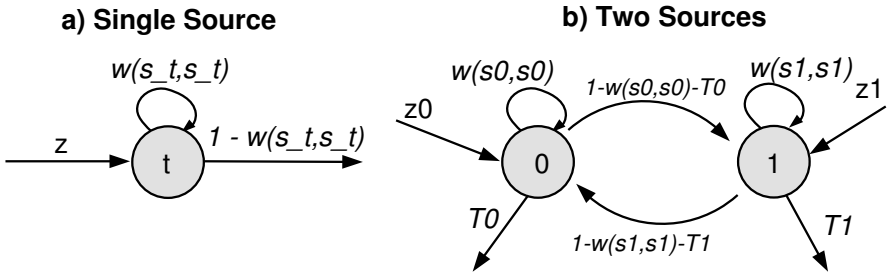

Figure 13: What is the Optimal Source Configuration?

a single *target source* through the manipulation of links (both from within the source and from other sources), which corresponds to the vulnerabilities identified in Section 3.1.

3.4.1 Link Manipulation Within a Source

We begin by studying link manipulation that is confined to a single source, which would correspond to collusive arrangements among spammer-controlled Web pages like link farms and link exchanges. In the source view of the Web, all intra-source page links are reflected in a single self-edge to the source, and all page links to others sources are reflected in source edges to external sources.

How should the Web spammer configure the target source s_t to maximize its SLA_{Rank} score, which in turn will have the greatest impact on the target source's rank relative to all other sources? In Figure 13(a), we consider a generic source configuration for s_t. The target has a self-edge weight of $w(s_t, s_t)$, leaving $1 - w(s_t, s_t)$ for all source edges to external sources. Let z denote the aggregate incoming score to the target source from sources beyond the control of the Web spammer. Here, the Web spammer has direct influence over its own links (reflected in $w(s_t, s_t)$) but no influence over the incoming links from other sources. As prescribed in Equation 1, we can write the target source's score:

$$\sigma_t = \alpha z + \alpha \cdot w(s_t, s_t) \cdot \sigma_t + \frac{1-\alpha}{|\mathcal{S}|}$$

$$\sigma_t = \frac{\alpha z + \frac{1-\alpha}{|\mathcal{S}|}}{1 - \alpha \cdot w(s_t, s_t)}$$

which is maximized when $w(s_t, s_t) = 1$. The optimal configuration is for the source to

eliminate all out edges and retain only a self-edge. Hence, the optimal σ_t^* is:

$$\sigma_t^* = \frac{\alpha z + \frac{1-\alpha}{|\mathcal{S}|}}{1-\alpha} \tag{3}$$

Impact of Influence Throttling: Given that the target source has an initial throttling factor $\kappa < 1$ and that $w(s_t, s_t) = \kappa$, the next question to consider is by how much may a source improve its score by adopting a self-edge weight even higher than its throttling factor (i.e., by increasing $w(s_t, s_t)$ beyond the mandated minimum κ throttling value)? Examining the relative SLA_{Rank} score for s_t, we have:

$$\frac{\sigma_t^*}{\sigma_t} = \frac{\frac{\alpha z + \frac{1-\alpha}{|\mathcal{S}|}}{1-\alpha}}{\frac{\alpha z + \frac{1-\alpha}{|\mathcal{S}|}}{1-\alpha\kappa}} = \frac{1-\alpha\kappa}{1-\alpha}$$

For a source with an initial baseline throttling value of $\kappa = 0$, a source may increase its SLA_{Rank} score by $\frac{1}{1-\alpha}$ by increasing its $w(s_t, s_t)$ to 1. For typical values of α – from 0.80 to 0.90 – this means a source may increase its score from 5 to 10 times. For sources that are more throttled there is less room for manipulation. In Figure 14, we show for increasing values of a baseline κ, the maximum factor change in SLA_{Rank} score by tuning the κ value closer to 1. A highly-throttled source may tune its SLA_{Rank} score upward by a factor of 2 for an initial $\kappa = 0.80$, a factor of 1.57 times for $\kappa = 0.90$, and not at all for a fully-throttled source.

By including self-edges in the source graph and the throttling factor κ, we allow a Web spammer some room for manipulating the score of its sources; however, the manipulation is for a *one-time* increase only and it may be limited by tuning the κ throttling factor higher. No such limit is provided under PageRank, meaning that a Web spammer may arbitrarily increase the score of a series of target pages by a factor even larger than we see for SLA_{Rank}.

3.4.2 Cross-Source Link Manipulation

We now study link manipulation across two or more sources, which corresponds to hijacking and honeypots scenarios, as well as collusive arrangements that span multiple sources. For this analysis, the spammer wishes to maximize the score for the single target source by manipulating the links available in one or more *colluding* sources.

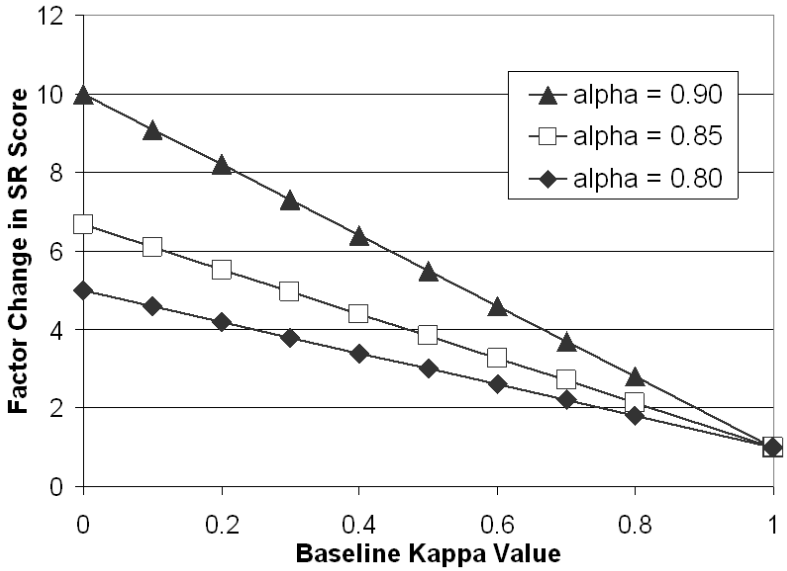

Figure 14: Change in Spam-Resilient SR Score By Tuning κ from a Baseline Value to 1

In Figure 13(b), we show a generic source configuration for a single target source s_0 and single colluding source s_1. We let θ_0 and θ_1 denote the edge weighting for each source to sources outside the sphere of influence of the Web spammer. Hence, source s_0 has $1 - w(s_0, s_0) - \theta_0$ edge weighting available for the edge to source s_1. The corresponding edge weight holds for the edge from s_1 to s_0. Let z_0 and z_1 denote the incoming score to each source, respectively, from other sources beyond the control of the Web spammer. Hence, we may describe the SLA_{Rank} for the two sources with a system of equations, where the Web spammer may manipulate $w(s_0, s_0)$, $w(s_1, s_1)$, θ_0, and θ_1:

$$\sigma_0 = \alpha z_0 + \alpha w(s_0, s_0)\sigma_0 + \frac{1-\alpha}{|\mathcal{S}|} + \alpha(1 - w(s_1, s_1) - \theta_1)\sigma_1$$
$$\sigma_1 = \alpha z_1 + \alpha w(s_1, s_1)\sigma_1 + \frac{1-\alpha}{|\mathcal{S}|} + \alpha(1 - w(s_0, s_0) - \theta_0)\sigma_0$$

Solving and taking the partial derivative with respect to the four parameters, we find that the optimal scenario for a Web spammer who wishes to maximize the SLA_{Rank} score for source s_0 is to set $\theta_0 = \theta_1 = 0$, meaning that there are no source edges to sources outside of the Web spammer's sphere of influence; $w(s_0, s_0) = 1$, meaning that the target source points only to itself and not at all to the colluding source; $w(s_1, s_1) = 0$, meaning that the

colluding source points only to the target source. With the κ_1 throttling factor requirement this means that the best the colluding source can do is meet the minimum requirement $w(s_1, s_1) = \kappa_1$ and direct the rest $(1 - \kappa_1)$ to the target.

If we extend this analysis to consider x colluding sources (labelled $s_1, ...s_x$) all in service to a single target source, then the system of equations is:

$$\sigma_0 = \alpha z_0 + \alpha w(s_0, s_0)\sigma_0 + \frac{1-\alpha}{|\mathcal{S}|} + \alpha \sum_{i=1}^{x} (1 - w(s_i, s_i)) \frac{\alpha z_i + \frac{1-\alpha}{|\mathcal{S}|}}{1 - \alpha w(s_i, s_i)}$$

$$\sigma_i = \alpha z_i + \alpha w(s_i, s_i)\sigma_i + \frac{1-\alpha}{|\mathcal{S}|} + \alpha(1 - w(s_0, s_0) - \theta_0)\sigma_0$$

The optimal configuration is for all colluding sources to set $\theta_i = 0$, meaning that there are no source edges from colluding sources to sources outside of the Web spammer's sphere of influence; $w(s_0, s_0) = 1$, meaning that the target source points only to itself and not at all to the colluding source; $w(s_i, s_i) = \kappa_i$, meaning that the colluding source directs the minimum edge weight to itself and the remainder $(1 - \kappa_i)$ to the target source. Hence, each colluding source s_i contributes some SLA_{Rank} $\Delta_{\sigma_i}(\sigma_0)$ to the target s_0:

$$\Delta_{\sigma_i}(\sigma_0) = \frac{\alpha}{1-\alpha} \sum_{i=1}^{x} (1 - \kappa_i) \frac{\alpha z_i + \frac{1-\alpha}{|\mathcal{S}|}}{1 - \alpha\kappa_i} \tag{4}$$

Clearly, tuning the κ throttling factor for each source closer to 1 (meaning that the majority of the colluding source's edge weight is directed to itself) results in a smaller change to the score of the target source. Hence, the introduction of the self-edge and the use of the throttling factor limits the impact of inter-source link manipulation.

Impact of Influence Throttling: To further understand the importance of the κ throttling factor on muting the impact of a Web spammer across sources, we consider a scenario in which a Web spammer controls x colluding sources, that each source has the same throttling factor of κ, and that the sources are configured optimally (as described above). Now suppose the throttling factor is raised to κ' for each source, meaning that each colluding source has less influence on the target source. How many sources x' are needed to achieve the same score as in the original case? I.e., what impact does raising the throttling factor have on the Web spammer?

If we let $z_i = 0$, we may write the original Spam-Resilient SLA_{Rank} score with x colluding sources and an initial throttling factor κ as well as the SLA_{Rank} score under the higher throttling factor (κ') scenario:

$$\sigma_0(x, \kappa) = \frac{(\frac{\alpha(1-\kappa)x}{1-\alpha\kappa} + 1)\frac{1-\alpha}{|\mathcal{S}|}}{1-\alpha}$$
$$\sigma_0(x', \kappa') = \frac{(\frac{\alpha(1-\kappa')x'}{1-\alpha\kappa'} + 1)\frac{1-\alpha}{|\mathcal{S}|}}{1-\alpha}$$

Letting $\sigma_0(x, \kappa) = \sigma_0(x', \kappa')$, we may find a relationship between the number of original colluding sources x and the number of colluding sources x' necessary under the higher throttling factor:

$$\frac{x'}{x} = \frac{1-\alpha\kappa'}{1-\alpha\kappa} \cdot \frac{1-\kappa}{1-\kappa'}$$

In Figure 15, we plot the percentage of additional sources ($\frac{x'}{x} - 1$) needed for a choice of κ' to equal the same influence on the score of the target page as under an initial choice $\kappa = 0$. For example, when $\alpha = 0.85$ and $\kappa' = 0.6$, there are 23% more sources necessary to achieve the same score as in the case when $\kappa = 0$. When $\kappa' = 0.8$, the Web spammer needs to add 60% more sources to achieve the same influence; for $\kappa' = 0.9$, he needs 135% more sources; and for $\kappa' = 0.99$, he needs 1485% more sources. Tuning the throttling factor higher considerably increases the cost of inter-source manipulation.

3.4.3 Comparison with PageRank

Now that we have studied source-centric ranking and seen how influence throttling can be used to significantly increase the cost of manipulation to a Web spammer, we next compare SLA_{Rank} to PageRank. Since PageRank provides page-level rankings, we consider a Web spammer whose goal is to maximize his influence over a single *target page* within a target source. Extending the framework from the previous section, we consider three scenarios (as illustrated in Figure 16).

1. The target page and all colluding pages belong to the same source.

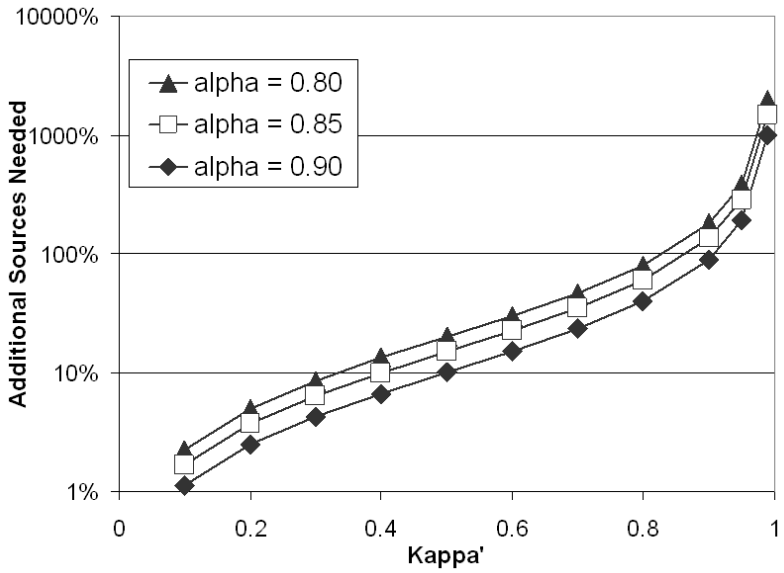

Figure 15: Additional Sources Needed Under κ' to Equal the Impact when $\kappa = 0$

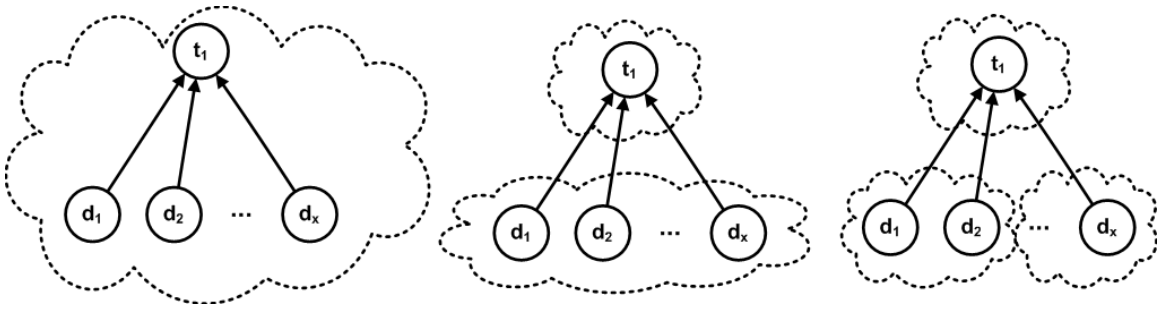

Figure 16: Three Link Manipulation Scenarios

2. The target page belongs to one source, and all colluding pages belong to one colluding source.

3. The target page belongs to one source, and the colluding pages are distributed across many colluding sources.

For each scenario, the colluding pages are structured with a single link to the target page. We consider the impact of an increasing number of colluding pages (τ). Adopting a linear formulation of PageRank that is similar in spirit to Equation 1, we may denote the PageRank score π_0 for the target page in terms of the PageRank due to pages outside of the sphere of influence of the Web spammer, the PageRank due to the teleportation component, and the PageRank due to the τ colluding pages:

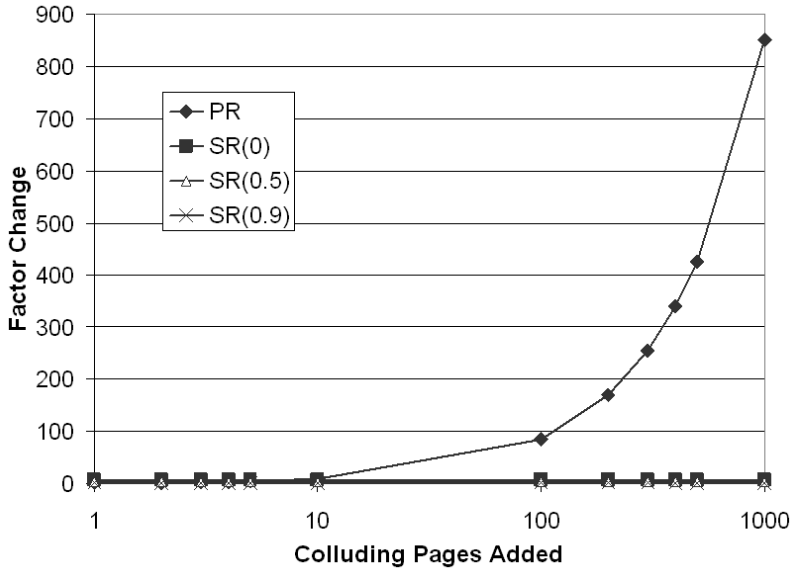

Figure 17: Comparison with PageRank: Scenario 1

$$\pi_0 = z + \frac{1-\alpha}{|\mathcal{P}|} + \tau\alpha\frac{1-\alpha}{|\mathcal{P}|}$$

where α refers to the teleportation probability and $|\mathcal{P}|$ refers to the total number of pages in the page graph. The contribution of the τ colluding pages to the overall PageRank score of the target page is:

$$\Delta_\tau(\pi_0) = \tau\alpha\frac{1-\alpha}{|\mathcal{P}|}$$

For Scenario 1, the Web spammer configures the target source optimally (as we presented in Equation 3), meaning that the colluding pages' intra-source links to the target page have no impact on the SLA_{Rank} score (other than perhaps a one-time increase due to tuning the self-edge weight up from κ to 1). PageRank, however, is extremely susceptible, as illustrated in Figure 17, where the PageRank score (PR) of the target page jumps by a factor of nearly 100 times with only 100 colluding pages.

For Scenario 2, the Web spammer adopts the optimal (worst-case) two-source configuration discussed in the previous section. In this configuration, the target source points only to itself, and the colluding source that contains the colluding pages directs κ edge weight to itself and the rest to the target source. In Figure 18, we see how PageRank is again

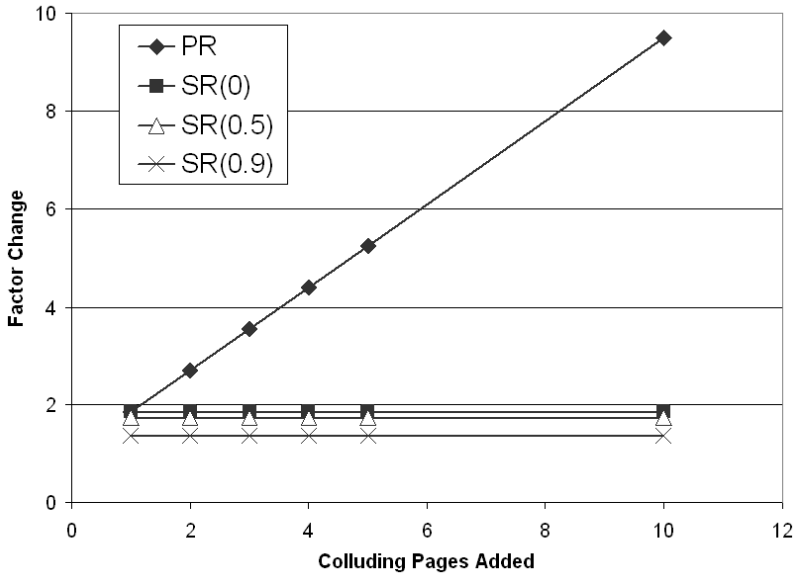

Figure 18: Comparison with PageRank: Scenario 2

extremely susceptible to such collusion, whereas the maximum influence over SLA_{Rank} is capped at 2 times the original score for several values of κ. Since PageRank has no notion of a source, makes no effort to regulate the addition of new pages to the Web graph, and has no notion of influence throttling, all three spam scenarios under consideration will have the same extreme impact on the PageRank score of the target page.

In Scenario 3, the Web spammer adopts the optimal configuration for x colluding sources (as we established in the previous section). Figure 19 plots the extreme impact on PageRank. As the influence throttling factor is tuned higher (up to 0.99), the SLA_{Rank} score of the target source is less easily manipulated.

3.5 Experimental Evaluation

In this section, we experimentally evaluate source-centric link analysis in the context of Web ranking with respect to four objectives – spam-resilience, time complexity, ranking stability, and approximating PageRank. Our spam-resilience evaluation is intended to confirm the analysis of the previous section over real Web data. We are interested to understand what are the most important parameters, how these parameters should be set, and what impact these parameter settings have on competing objectives. We find that careful tuning of these

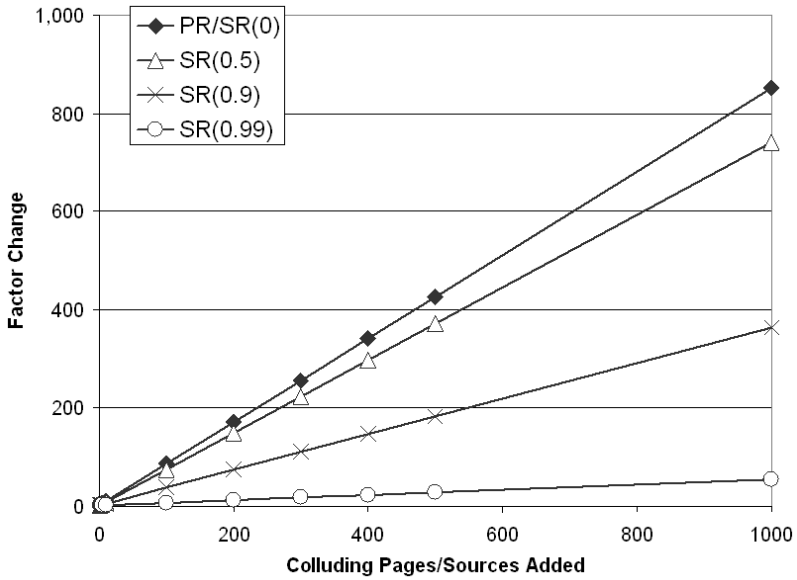

Figure 19: Comparison with PageRank: Scenario 3

Table 4: Source Graph Summary – By Source Definition

	Domain		Host		Dir		Rand		Page	
Dataset	**Nodes**	**Links**	**Nodes**	**Links**	**Nodes**	**Links**	**Nodes**	**Links**	**Nodes**	**Links**
UK2002	81k	1.2m	98k	1.6m	360k	3.5m	98k	286.0m	18.5m	292.2m
IT2004	136k	2.7m	141k	2.8m	505k	8.6m	141k	1,069.3m	41.3m	1,135.7m
WB2001	620k	10.5m	739k	12.4m	3,315k	24.7m	739k	955.4m	118.1m	992.8m

parameters is vital to ensure success over each objective, and that some objectives cannot be maximized without negatively impacting other objectives.

3.5.1 Experimental Setup

All of our experimental evaluation is over the three Web datasets described in Section 3.2.2. For each dataset we extracted the domain, host, and directory information for each page URL and assigned pages to sources based on these characteristics. We also consider the extreme case when each page belongs to its own source (equivalent to the page graph described in Table 2). For the random source definition, we set the number of nodes in the graph to be the same as the number of hosts. In Table 4, we present summary information for each of the source graphs (including self-edges). Note that the random source graph displays nearly no link-locality, with less than 1% of all page links being intra-source.

All of the ranking code was written in Java. The data management component was

based on the WebGraph compression framework described in [26] for managing large Web graphs in memory. All experiments were run on a dual processor Intel XEON at 2.8GHz with 8GB of memory. We measured the convergence rate for all ranking calculations using the L2-distance of successive iterations of the Power Method. We terminated the ranking calculations once the L2-distance dropped below a threshold of 10e-9.

As a baseline, we computed the global PageRank vector ($\boldsymbol{\pi}$) over each page graph using the standard Power Method and the parameters typically used in the literature (e.g., [138]), including a mixing parameter of 0.85, a uniform teleportation vector, and a uniform link following probability.

For the quality-weighted edge weighting, we measure the quality of each page $q(p_i)$ using the page's PageRank score π_i. Although in practice it may not be reasonable to use the PageRank score as a measure of quality since it is so expensive to calculate, we include these PageRank-weighted options here to more fully understand their impact relative to the edge weighting schemes that do not require PageRank.

For compactness, we shall write a particular SLA_{Rank} parameter combination like $\mathcal{SR}(\mathbf{T}^*_U, \mathbf{c}_u)$, where the transition matrix $\mathbf{T}$ is appended with a subscript to indicate which source edge weighting scheme we use: $\mathbf{T}_U$, $\mathbf{T}_{LC}$, and so on. We shall append an asterisk to the transition matrix to indicate the inclusion of self-edges: $\mathbf{T}^*$. For the choice of teleportation vector $\mathbf{c}$, we consider the standard uniform vector ($\mathbf{c}_u$) and the source-size-based vector ($\mathbf{c}_\mathbf{s}$).

3.5.2 Measures of Ranking Distance

We rely on two distance metrics for comparing ranking vectors. The Kendall Tau Distance Metric [59] is based solely on the relative ordering of the sources in two ranking vectors. In contrast, the Jensen-Shannon Divergence [109] measures the distributional similarity of two vectors, meaning that it considers the magnitude of each source's authority score and not just the relative ordering of the sources.

Kendall Tau Distance Metric. This metric measures the relative ordering of two lists of ranked objects [59]. It is based on the original Kendall Tau correlation described in [99]

and provides a notion of how closely two lists rank the same set of objects (or Web sources in our case). The Kendall Tau distance metric takes values in the range [0,1], where two rankings that are exactly the same have a distance of 0, and two rankings in the reverse order have a distance of 1. We rely on a variation of an $O(n \log n)$ version described in [25].

JS-Divergence. The Jensen-Shannon divergence is a measure of the distributional similarity between two probability distributions [109]. It is based on the relative entropy measure (or KL-divergence), which measures the difference between two probability distributions p and q over an event space X: $KL(p,q) = \sum_{x \in X} p(x) \cdot \log(p(x)/q(x))$. If we let p be one of the ranking vectors $\boldsymbol{\sigma}$, and q be the other ranking vector $\boldsymbol{\sigma'}$, then we have $KL(\boldsymbol{\sigma}, \boldsymbol{\sigma'}) = \sum_{i \in \mathcal{S}} \sigma_i \cdot \log(\sigma_i/\sigma'_i)$. Intuitively, the KL-divergence indicates the inefficiency (in terms of wasted bits) of using the q distribution to encode the p distribution. Since the KL-divergence is not a true distance metric, the JS-divergence has been developed to overcome this shortcoming, where:

$$JS(\boldsymbol{\sigma}, \boldsymbol{\sigma'}) = \phi_1 KL(\boldsymbol{\sigma}, \phi_1 \boldsymbol{\sigma} + \phi_2 \boldsymbol{\sigma'}) + \phi_2 KL(\boldsymbol{\sigma'}, \phi_1 \boldsymbol{\sigma} + \phi_2 \boldsymbol{\sigma'})$$

and $\phi_1, \phi_2 > 0$ and $\phi_1 + \phi_2 = 1$. In the experiments reported in this chapter, we consider $\phi_1 = \phi_2 = 0.5$. The JS-divergence takes values in the range [0,1] with lower values indicating less distance between the two ranking vectors.

3.5.3 Objective-Driven Evaluation

We report the most significant results from a total of 360 different ranking vectors we computed by combining the five source definitions, the 12 source-citation edge weights, the two teleportation vectors, and the three datasets. For the 360 ranking vectors we analyze, we fix the mixing parameter α at the commonly adopted value of 0.85 used for PageRank (e.g., [138]). Note that we also varied α in our preliminary experiments, but find that there is no significant change in the results we report here.

Dataset	Source Definition				
	Domain	**Host**	**Dir**	**Rand**	**Page**
UK2002	0.02	0.03	0.07	2.45	2.76
IT2004	0.05	0.05	0.13	9.33	9.44
WB2001	0.21	0.25	0.46	11.32	12.28

Table 5: Wallclock Time (Minutes) Per Iteration

3.5.4 Time Complexity

We begin by examining the ranking efficiency of the source-centric ranking approach in terms of the time it takes to calculate each ranking vector. The PageRank-style calculation scans the link file for each source graph multiple times until convergence.

Table 5 shows the average time per iteration to calculate the ranking vector over the five different source graphs for each of the datasets. We report the results for a ranking based on the uniform edge weight and a uniform teleportation vector: $\mathcal{SR}(\mathbf{T}_U, \mathbf{c}_u)$. These general per-iteration results also hold for the total time to reach the L2 stopping criterion. In our examination of the 360 different ranking vectors, we find that the source definition has the most significant impact on the calculation time, since the source definition directly impacts the size of the source graph. The choice of edge weights and teleportation vector has little discernable impact on the calculation time.

As we can see, the directory, host, and domain source definitions result in ranking computation that is one to two orders of magnitude faster than the page-based graph. Since PageRank over a Web graph of billions of nodes takes days or weeks, this improvement is important for source-centric ranking to compensate for PageRank's slow time-to-update. The random source definition performs poorly, even though there are the same number of nodes in the random graph as in the host graph. The key difference is that the random graph has no link locality structure, and hence consists of nearly as many links as in the page graph. We conclude that link locality strongly impacts the degree of source graph size reduction, and hence, the ranking calculation time. Due to its poor performance, we shall drop the random source definition from the rest of our reported experimental results.

Shorthand	Version	Edge Weight	Self-Edges?	Telep. Factor
Baseline	$\mathcal{SR}(\mathbf{T}^*_{LC}, \mathbf{c}_u)$	LC	Yes	Uniform
NoL	$\mathcal{SR}(\mathbf{T}_{LC}, \mathbf{c}_u)$	LC	No	Uniform
Size	$\mathcal{SR}(\mathbf{T}^*_{LC}, \mathbf{c}_s)$	LC	Yes	Size
Uni	$\mathcal{SR}(\mathbf{T}^*_{U}, \mathbf{c}_u)$	U	Yes	Uniform
SC	$\mathcal{SR}(\mathbf{T}^*_{SC}, \mathbf{c}_u)$	SC	Yes	Uniform
TD	$\mathcal{SR}(\mathbf{T}^*_{TD}, \mathbf{c}_u)$	TD	Yes	Uniform
LC(q)	$\mathcal{SR}(\mathbf{T}^*_{LC(q)}, \mathbf{c}_u)$	LC(q)	Yes	Uniform
SC(q)	$\mathcal{SR}(\mathbf{T}^*_{SC(q)}, \mathbf{c}_u)$	SC(q)	Yes	Uniform

Table 6: Ranking Similarity: Parameter Settings

3.5.5 Stability – Ranking Similarity

We next explore the parameter space to investigate the stability in terms of the similarity of rankings induced by the various parameter settings. Due to its popularity in other works (e.g. [97, 181]), we adopt a baseline ranking based on the link count edge weight with self-edges and a uniform teleportation vector, $\mathcal{SR}(\mathbf{T}^*_{LC}, \mathbf{c}_u)$, and report seven alternative ranking vectors computed by tweaking these baseline parameter settings. We consider a version without self-edges ($\mathcal{SR}(\mathbf{T}_{LC}, \mathbf{c}_u)$), a version including self-edges and the size-based teleportation component ($\mathcal{SR}(\mathbf{T}^*_{LC}, \mathbf{c}_s)$), and five additional versions using the other edge weighting schemes (e.g., $\mathcal{SR}(\mathbf{T}^*_{U}, \mathbf{c}_u)$) as shown in Table 6. We report the results for the host-based graph in this section; we see similar results across the directory and domain source definition settings.

In Figures 20, 21, and 22 we compare the ranking vector resulting from the baseline parameter settings with the ranking vector resulting from each of these seven alternative parameter settings. The y-axis measures the distance between these alternative ranking vectors and the baseline configuration via the Kendall Tau Distance Metric and the JS-Divergence.

As we can see, the exclusion of self-edges (`NoL`) and the choice of teleportation vector (`Size`) are the two factors with the most significant impact on the resulting ranking vector in terms of ranking distance from the baseline setting. Hence, we must be careful when setting these two critical parameters, since the resulting ranking vectors depend so heavily on

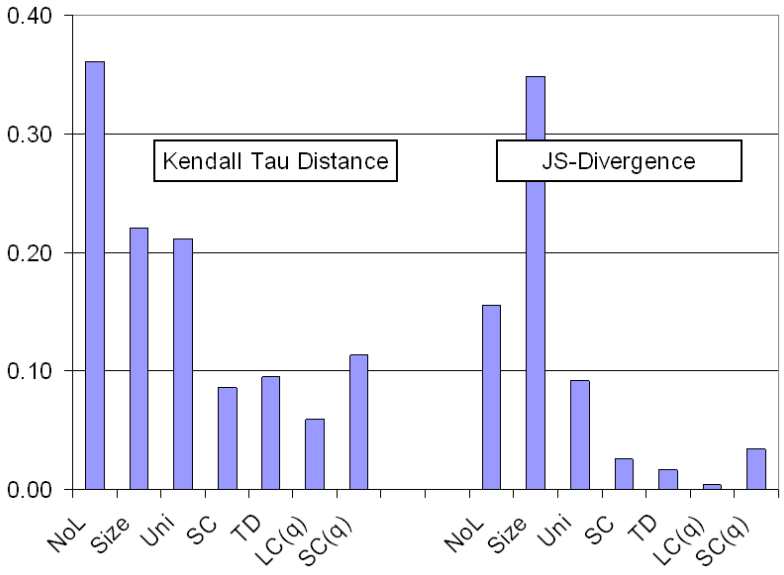

Figure 20: Parameter Tuning: Ranking Distance Versus Baseline Configuration, UK2002

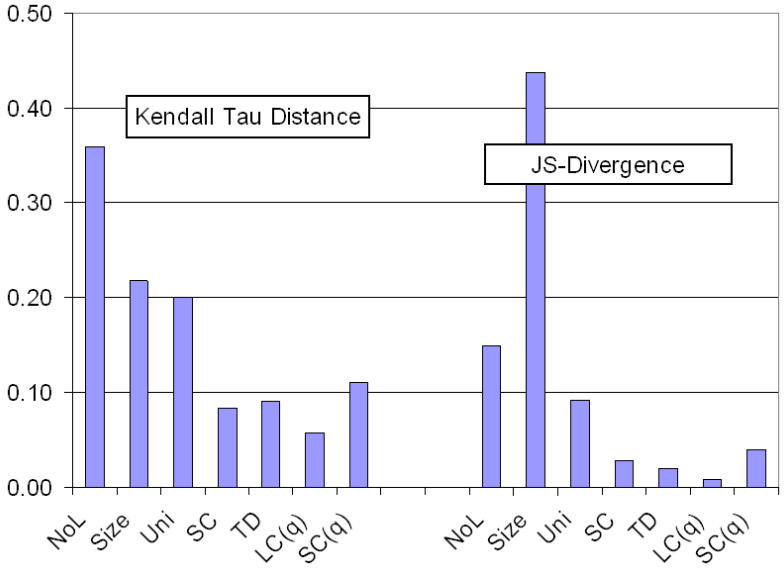

Figure 21: Parameter Tuning: Ranking Distance Versus Baseline Configuration, IT2004

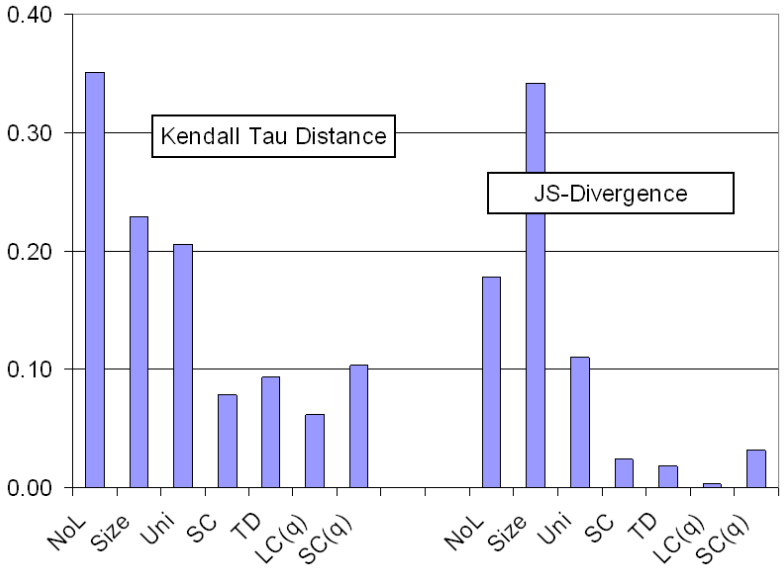

Figure 22: Parameter Tuning: Ranking Distance Versus Baseline Configuration, WB2001

them. The choice of edge weights has less impact, though we observe that the uniform edge weighting results in the most dissimilar ranking vector of all the edge weighting schemes. The uniform edge weighting scheme is a less intuitively satisfactory edge weighting scheme, and these results confirm this view. What is interesting here is that the Source Consensus, Target Diffusion, Quality-Weighted Link Count, and Quality-Weighted Source Consensus edge weights have a relatively minor impact on the resulting ranking vector versus the baseline Link Count version. We note that the Quality-Weighted Link Count deviates very little from the Link Count version, in spite of the incorporation of the expensive PageRank scores. This is encouraging since it means we need not rely on PageRank for assessing source quality.

3.5.6 Stability – Link Evolution

We next evaluate the stability of SLA_{Rank} as the Web graph evolves for each of the three Web datasets. Since the source view of the Web provides an aggregate view over Web pages, we anticipate that domain, host, and directory-based rankings should be less subject to changes in the underlying page graph than page-based rankings. Our goal is to emulate the gradual discovery of Web pages, similar to how a Web crawler may incrementally discover

new pages for ranking.

For each dataset, we randomly selected a fraction of the pages (10%, 30%, ...) and computed the standard PageRank vector over just this fraction of pages, yielding $\boldsymbol{\pi}_{10\%}, \boldsymbol{\pi}_{30\%}$, and so on. Additionally, we computed the ranking vector for the domain, host, and directory-based source graphs derived from the same fraction of all pages, yielding $\boldsymbol{\sigma}_{10\%}, \boldsymbol{\sigma}_{30\%}$, and so on. We then compared the relative page rankings for the pages in $\boldsymbol{\pi}_{10\%}$, $\boldsymbol{\pi}_{30\%}$, ..., to the relative rankings of the *exact same pages* in the PageRank vector for the full Web page graph. Similarly, we compared the relative source rankings for the sources in $\boldsymbol{\sigma}_{10\%}, \boldsymbol{\sigma}_{30\%}, ...$, to the relative rankings of the *exact same sources* in the ranking vector for the full Web source graph. To evaluate the stability, we rely on the Kendall Tau Distance metric as a measure of ranking error.

In Figure 23 we show the ranking error for the WB2001 dataset for PageRank and for three representative parameter settings over the host-based source graph – the baseline version $\mathcal{SR}(\mathbf{T}^*_{LC}, \mathbf{c}_u)$, the loop-less version $\mathcal{SR}(\mathbf{T}_{LC}, \mathbf{c}_u)$, and the size-based teleportation version $\mathcal{SR}(\mathbf{T}^*_{LC}, \mathbf{c}_s)$. Note that these are the three settings that resulted in the most different ranking vectors in our previous experiment. In all cases, the source-centric rankings display significantly less error relative to the rankings over the full Web graph than the PageRank rankings do, meaning that we can rely on source-centric rankings computed over an incomplete Web crawl with substantial confidence. Also note that the size-based version is the most stable, and we find that this stability generally improves as the source definition becomes more inclusive (from page to directory to host to domain).

Since the page and source ranking vectors are of different lengths, we additionally considered a similar stability analysis over just the top-100 and top-1000 page and source rankings. We relied on a variation of the Kendall Tau Distance metric known as the Kendall Min Metric [59] for evaluating top-k ranked lists. These results further validate the source stability.

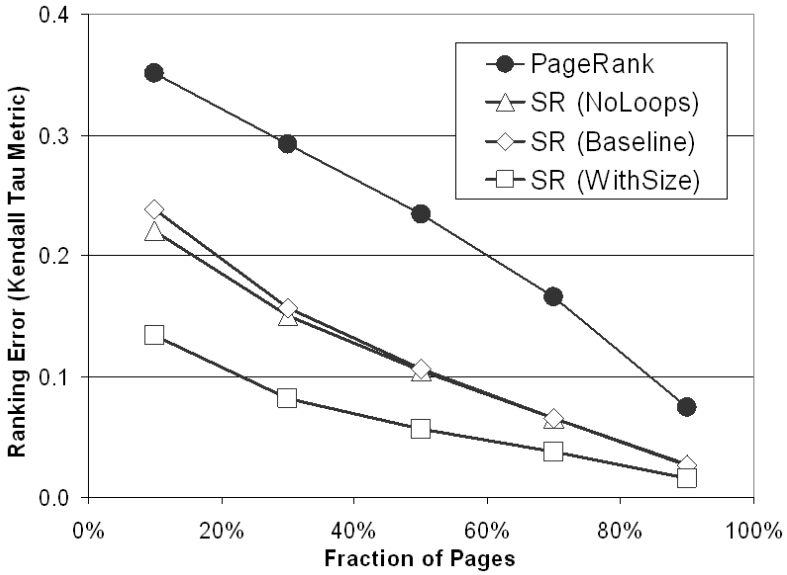

Figure 23: Ranking Stability, WB2001

3.5.7 Approximating PageRank

As we have mentioned, one of the important goals of source-centric ranking is to approximate the traditional global PageRank vector by combining the source-level ranking information with per-source ranking information (the local PageRank scores). Such approximation promises to speed the PageRank calculation considerably. In this experiment we aim to understand under what conditions source-centric ranking may be used to reasonably approximate PageRank.

To approximate PageRank, we decompose the global PageRank of a page into source and local components:

$$\pi(p_i) = \sigma(s_j) \cdot \pi(p_i|s_j) \tag{5}$$

where we denote the local PageRank score for page i in source j as $\pi(p_i|s_j)$. The local PageRank score is calculated based only on local knowledge (e.g., based on the linkage information of pages within the source), takes comparably little time relative to the full PageRank calculation, and forms a probability distribution (i.e., $\sum_{p_k \in s_j} \pi(p_k|s_j) = 1$).

For the PageRank decomposition to hold over all pages, ideally we would have that the

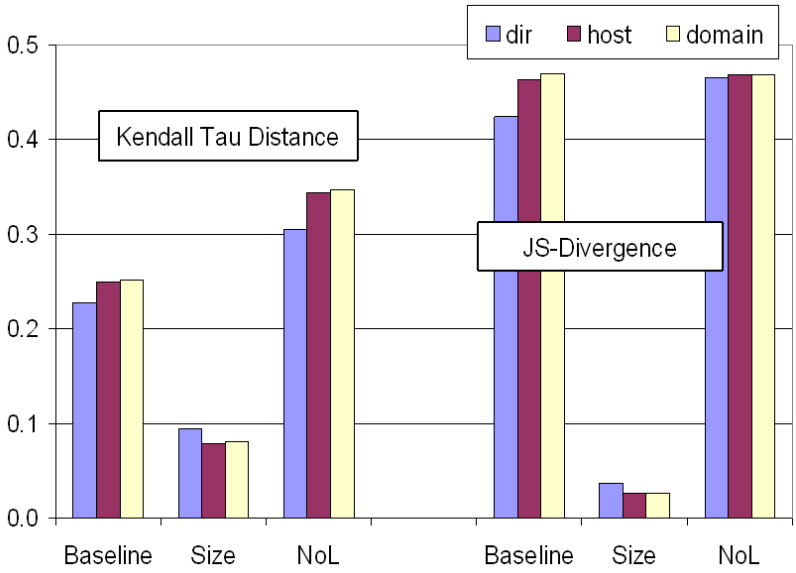

Figure 24: Approximating PageRank, IT2004

local PageRank scores $\pi(p_i|s_j)$ would exactly match the relative global distribution:

$$\pi(p_i|s_j) = \frac{\pi(p_i)}{\sum_{p_k \in s_j} \pi(p_k)} \tag{6}$$

By replacing $\pi(p_i|s_j)$ in Equation 5 with the righthand-side of Equation 6, we find that the source-centric component $\sigma(s_j)$ should equal the sum of the global PageRanks of the constituent pages:

$$\sigma(s_j) = \sum_{p_k \in s_j} \pi(p_k)$$

To test how well the source-centric rankings may be used to approximate PageRank, we compare the rankings induced from various parameter settings with the rankings induced from ranking the sources by the sum of the global PageRanks of their constituent pages. In Figure 24, we report the Kendall Tau Distance Metric and the JS-Divergence for three representative parameter settings – the baseline version $\mathcal{SR}(\mathbf{T}^*_{LC}, \mathbf{c}_u)$, the loop-less version $\mathcal{SR}(\mathbf{T}_{LC}, \mathbf{c}_u)$, and the size-based teleportation version $\mathcal{SR}(\mathbf{T}^*_{LC}, \mathbf{c}_s)$ – over the domain, host, and directory-based source definitions for the IT2004 dataset. Similar results hold for the other two datasets.

The baseline parameter setting (which has been used elsewhere, e.g., [97, 181]) performs poorly, and is not appropriate for approximating PageRank. Similarly, the loopless version, which disregards the strong evidence of link-locality for setting edge weights, also performs poorly. Only the size-based version is highly correlated with the sum of the actual PageRank values for each source, meaning that source size and the presence of self-edges are critical for approximating PageRank. We also find that high-quality results hold when we replace the link count edge weighting parameter with the source consensus and target diffusion schemes.

3.5.8 Spam-Resilience

Finally, we study the spam-resilience properties of source-centric link analysis through three popular Web spam scenarios.

3.5.8.1 Spam Scenario 1: Intra-Source Link Farm

We first aim to validate the analysis in Section 3.4.1 by considering the impact of page-level manipulation *within* a single source. For this *intra-source manipulation* we study the impact of a spammer who manipulates the pages internal to a target source for increasing the rank of a target page within the target source. We again consider the three representative parameter settings – the baseline, loop-less, and size-based teleportation versions. For each version, we randomly selected five sources from the bottom 50% of all sources on the host graph. For each source, we randomly selected a target page within the source and created a link farm consisting of a single new spam page within the same source with a link to the target page. This is case A. For case B, we added 10 spam pages to the link farm within the source, each with a link to the target page. We repeated this setup for 100 pages (case C) and 1,000 pages (case D). For each case, we then constructed the new spammed page graph and host graph for each of the three Web datasets. We ran PageRank and the three source-centric versions for each of the four cases. In Figures 25, 26, and 27 we show the influence of the Web spammer in manipulating the rank of the target page and the rank of the target source through the average ranking percentile increase. For example in the WB2001 case, the PageRank of the target page jumped 80 percentile points under case C

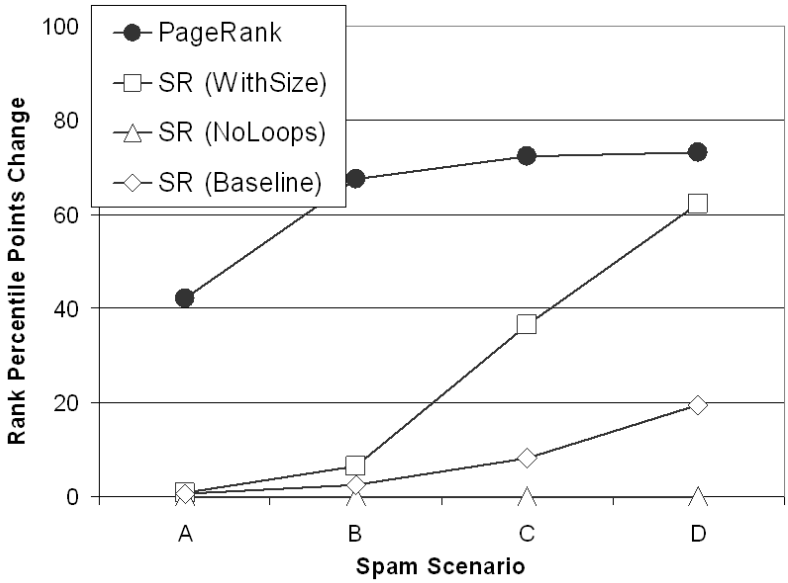

Figure 25: Intra-Source Link Farm, UK2002

(from an average rank in the 19th percentile to the 99th), whereas the score of the target source jumped only 4 percentile points for the baseline version (from an average rank in the 27th percentile to the 31st).

We first note the dramatic increase in PageRank for the target page across all three Web datasets, which confirms the analysis about the susceptibility of PageRank to rank manipulation. Although PageRank has typically been thought to provide fairly stable rankings (e.g., [131]), we can see how link-based manipulation has a profound impact, even in cases when the spammer expends very little effort (as in cases A and B). We note that the loop-less source-centric version shows no change in rank value, since the addition of new intra-source links has no impact on the resulting source graph and ranking vector. The baseline version does increase some, but not nearly as much as PageRank. Since the source is an aggregation of many pages, the weighting of the source edges is less susceptible to changes in the underlying page graph. In contrast, the size-based teleportation version is the most vulnerable to intra-source manipulation. In fact, under this scenario, a spammer need only add new pages (not links) to increase the score of a source. The addition of so many new pages increases the size of the source, making it more attractive to the random walker who considers the size-based teleportation component. In fact, under this scenario,

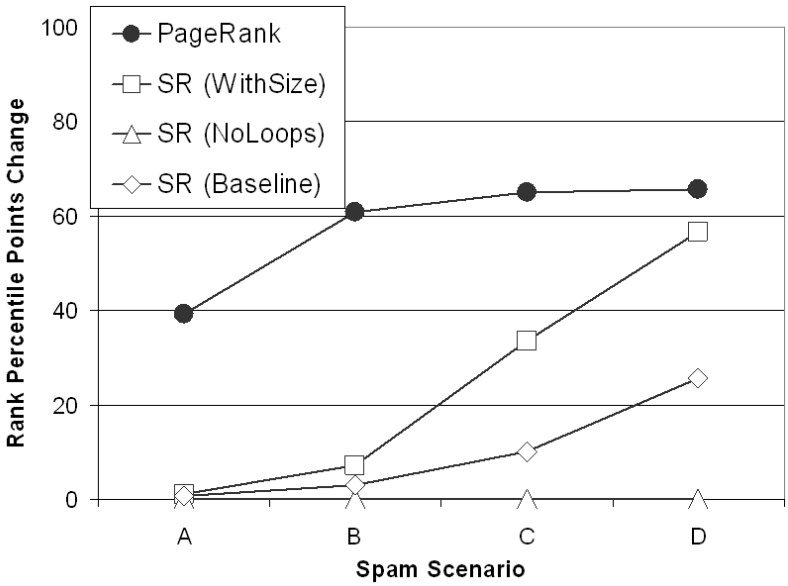

Figure 26: Intra-Source Link Farm, IT2004

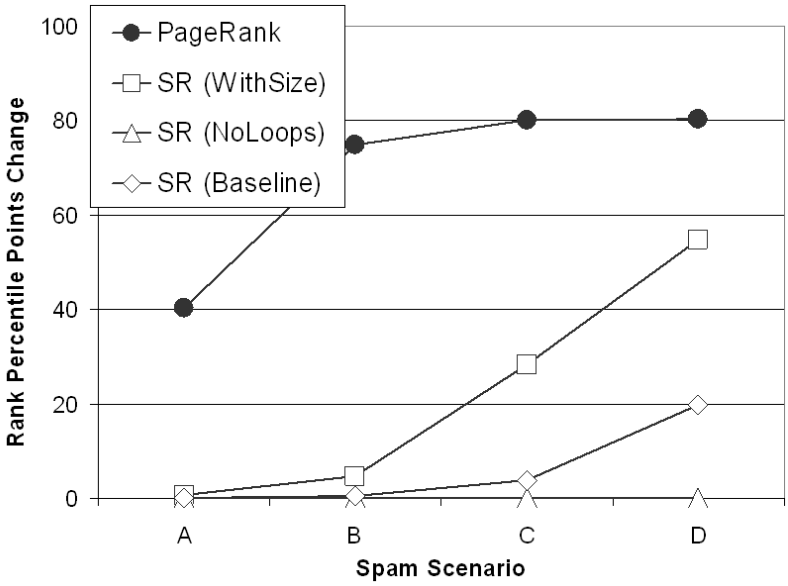

Figure 27: Intra-Source Link Farm, WB2001

a spammer need only add new pages (not links) to increase the score of a source.

3.5.8.2 Spam Scenario 2: Inter-Source Link Farm

In the second Web spam scenario, we consider the impact of manipulation *across* sources, which corresponds to the analysis in Section 3.4.2. For this scenario, the spam links are added to pages in a colluding source that point to the target page in a different source. We paired the randomly selected target sources from the previous experiment with a randomly selected colluding source, again from the bottom 50% of all sources. For each pair, we added a single spam page to the colluding source with a single link to the randomly selected target page within the target source. This is case A. We repeated this setup for 10 pages (case B), 100 pages (case C), and 1,000 pages (case D). For each case, we then constructed the new spammed page graph and source graph for each of the three Web datasets. We ran PageRank and Spam-Resilient SourceRank for each of the four cases.

In Figures 28, 29, and 30 we show the influence of the Web spammer in manipulating the rank of the target page and the target source. Since the page-level view of the Web does not differentiate between intra-source and inter-source page links, we again see that the PageRank score dramatically increases, whereas the source-centric scores are impacted less. We are encouraged to observe that all three source-centric versions perform better than PageRank. We witness this advantage using no additional influence throttling information for the sources under consideration, meaning that the source-centric advantage would be even greater with the addition of more throttling information. The baseline version does increase some, but not nearly as much as PageRank. Since the source is an aggregation of many pages, the weighting of the source edges is less susceptible to changes in the underlying page graph. Interestingly, the loop-less version is the least resistant to manipulation for cases A, B, and C. In the loop-less version, external links are the sole determiner of a source's rank, meaning that inter-source manipulation wields more influence than for the looped versions. The size-based teleportation version is the most vulnerable for case D.

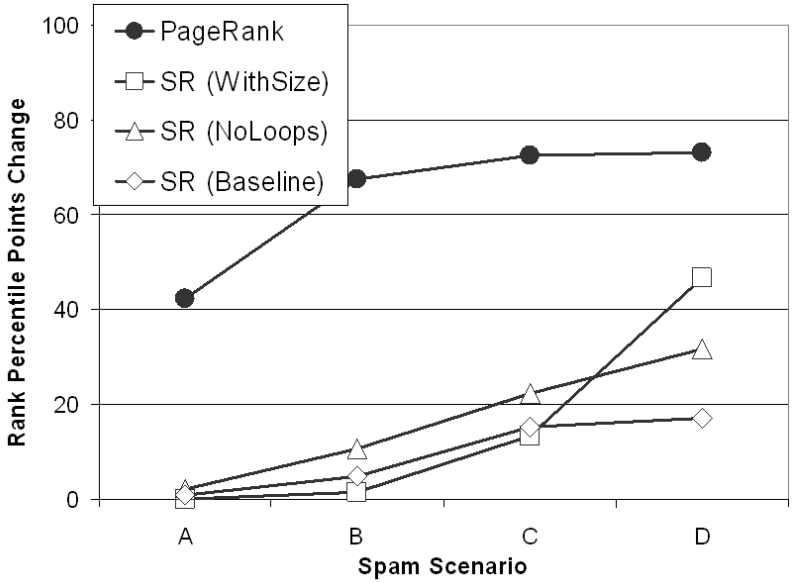

Figure 28: Inter-Source Link Farm, UK2002

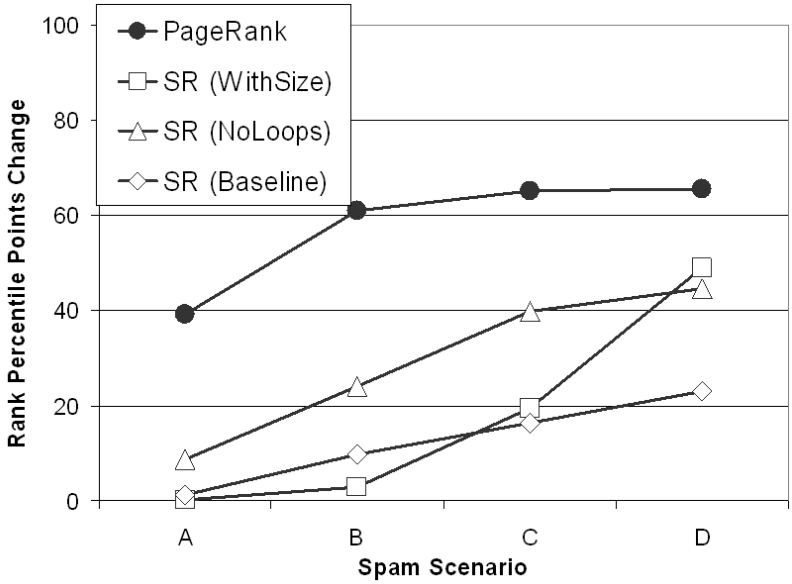

Figure 29: Inter-Source Link Farm, IT2004

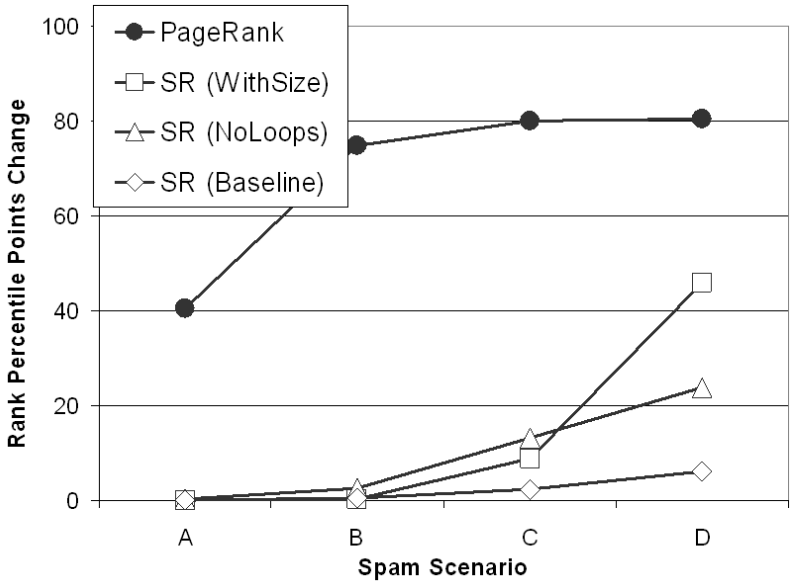

Figure 30: Inter-Source Link Farm, WB2001

3.5.8.3 Spam Scenario 3: Link Hijacking

In the third scenario, we consider a spammer who hijacks links from high-quality pages. The goal of link hijacking is to insert links into reputable pages that point to a spammer's target page, so that it appears to the ranking algorithm that the reputable page endorses the spam page. Spammers have a number of avenues for hijacking legitimate pages, including the insertion of spam-links into public message boards, openly editable wikis, and the comments section of legitimate Weblogs.

Given the results of the link farm scenario, we select the baseline parameter settings $\mathcal{SR}(\mathbf{T}^*_{LC}, \mathbf{c}_u)$, as well as two other versions based on the source consensus and uniform edge weights: $\mathcal{SR}(\mathbf{T}^*_{SC}, \mathbf{c}_u)$ and $\mathcal{SR}(\mathbf{T}^*_{U}, \mathbf{c}_u)$. For each version, we selected a target source ranked 100,000th on the WB2001 host graph and four different high-quality sources to hijack, each ranked close to 100th overall. Each hijacked source has different size and linking characteristics – case A is a large source with 3,000 pages and 69 source edges; case B is a medium source with 1,557 pages and 78 source edges; case C is a source with 416 pages and 336 source edges; finally, case D is a small source with 24 pages and 5 source edges.

For each hijacked source, we assume the hijacker has full control of only a single hijacked

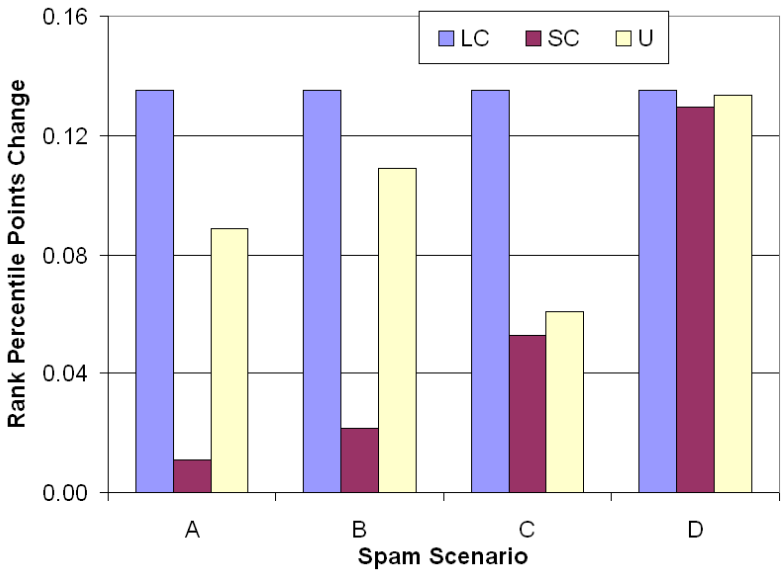

Figure 31: Link Hijacking, WB2001

page. Since the link count edge weight simply counts the number of links from one source to another, the spammer can insert sufficient links into the single hijacked page to skew the weight from the hijacked source to the target source to be nearly 1. In contrast, the uniform edge weight is constrained by the number of distinct targets (source edges). The source consensus edge weight is constrained by the number of hijacked pages.

In Figure 31, we show the influence of the link hijacker over the rank of the target source through the average ranking percentile increase. In all cases, the target source is originally in the 86th percentile (ranked 100k out of 739k). In case A, the target source jumps nearly 14 percentile points under the link count edge weight up to a final rank in the top-150 out of 738,626 sources. In contrast, the source consensus only jumps around 1 percentile points to a rank of 92k. As we can see, the source consensus performs the best across the four tested cases. We see that the link count approach, which is popular in other works, is extremely susceptible to hijacking across all cases. We also considered a second link hijacking scenario, in which a spammer inserts multiple links to many different target pages; we find that this behavior severely negatively impacts the target diffusion edge weighting, resulting in results nearly as poor as for the link count approach.

For both the link farm and link hijacking scenarios, we find that adopting either a

directory-based or domain-based source definition versus the host-based definition reported here leads to mixed results. In the directory-based source graph, sources are on-average smaller, meaning that the edge weight manipulation for each source is easier to achieve. But since there are more directories than hosts, a greater degree of manipulation is necessary to equal the same ranking boost as in the host graph. The opposite situation holds for the domain source definition, where edge weight manipulation is more difficult (since sources are larger), but less manipulation is necessary since there are fewer domains than hosts. We are continuing to study this phenomenon in our ongoing research.

3.5.9 Influence Throttling Effectiveness

In the final experiment, we study the impact of influence throttling on the spam-resilience characteristics of source-centric ranking. For the WB2001 dataset we manually identified 10,315 pornography-related sources, and labeled these as spam. It is unreasonable for a spam identification algorithm (whether manual or automated) to identify all spam sources with high precision. Hence, of these 10,315 spam sources, we randomly selected just 1,000 (fewer than 10%) to use as a seed set for the spam-proximity calculation. We calculated the spam-proximity score for each source using the approach described in Section 3.2.5.

Based on these scores, we assigned an appropriate throttling value to each source, such that sources that are "closer" to spam sources are throttled more than more distant sources. These spam proximity scores are propagated to all sources in the dataset based only on the seed set of 1,000 identified spam sources. We assigned the top-20,000 spam-proximity sources a throttling value of $\kappa = 1$, meaning that their influence was completely throttled. For all other sources we assigned a throttling value of $\kappa = 0$, meaning that these sources were throttled not at all. We then computed the source-centric ranking vector using these throttling values. As a point of comparison, we also computed the baseline ranking vector using no throttling information.

For each of the two ranking vectors, we sorted the sources in decreasing order of scores and divided the sources into 20 buckets of equal number of sources. Along the x-axis of

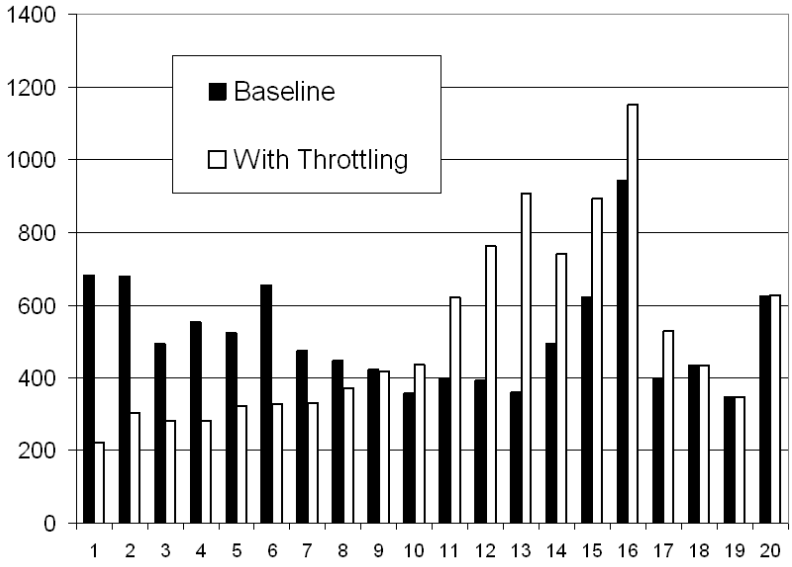

Figure 32: Rank Distribution of All Spam Sources

Figure 32 we consider these 20 buckets for the WB2001 dataset, from the bucket of top-ranked sources (bucket 1) to the bucket of the bottom-ranked sources (bucket 20). Along the y-axis, we plot the number of actual spam sources (of the 10,315 total spam sources) in each bucket. The approach using influence throttling penalizes spam sources considerably more than the baseline approach, even when fewer than 10% of the spam sources have been explicitly marked as spam.

3.5.10 Summary of Experiments

The evaluation of the parameterized source-centric link analysis framework has yielded several interesting observations:

- Source-centric link analysis heavily depends on the source definition and the degree of link locality. We find that a lack of locality results in poor time complexity, but that even moderate locality (e.g., $\sim 65\%$) leads to good time complexity and stability results that are comparable with source definitions that display extremely high locality.

- In terms of ranking vector stability across parameters, the most important parameters are self-edges and the source-size teleportation component. We also found that incorporating expensive quality information into the edge weighting schemes resulted

in only a slight change to the resulting ranking vector.

- To best approximate PageRank using a layered calculation and for the most stable rankings in the face of Web link evolution, we saw the critical need for using the size-based teleportation component.

- However, using the size-based teleportation component resulted in the most severe vulnerability to spam, although it has these two desirable properties. This fundamental tension in objectives motivates our continuing research.

- Finally, we saw how incorporating influence throttling information resulted in better spam-resilience properties than the baseline approach.

3.6 Related Work

In addition to the related work cited elsewhere in this chapter, there have been some other efforts to understand higher-level Web abstractions. In [52], the *hostgraph* was explored in terms of various graph properties like indegree and outdegree distribution, and size of connected components. Crawling mechanisms based on the site paradigm, rather than the traditional page-based one, were enumerated in [58]. In [57], the potential spam properties of a HostRank algorithm were observed, and in [167] the ranking quality of several site-level-style PageRank variations was studied. In contrast to page aggregations, other researchers [33] have considered dis-aggregating Web pages into smaller units for providing ranking over individual components of Web pages.

Source-centric ranking can also take advantage of algorithmic enhancements for speeding PageRank (e.g., [122]).

As we have noted, several studies have identified large portions of the Web to be subject to malicious rank manipulation [61, 80], especially through the construction of specialized link structures for promoting certain Web pages. In Chapter 2, we identified some techniques for dealing with link-based vulnerabilities. In addition, several researchers have studied collusive linking arrangements with respect to PageRank, including [192] and [12]. Link farms have been studied in [2]. Separately, optimal link farms and the effectiveness of spam

alliances have been studied in [78]. Davison [49] was the first to investigate the identification of so-called nepotistic links on the Web.

3.7 Summary

In this chapter, we have introduced a parameterized framework for source-centric link analysis, explored several critical parameters, and conducted the first large-scale comparative study of source-centric link analysis over multiple large real-world Web datasets and multiple competing objectives. We find that careful tuning of these parameters is vital to ensure success over each objective and to balance the performance across all objectives. We have introduced the notion of influence throttling, studied analytically its impact, and provided experimental validation of the effectiveness and robustness of our spam-resilient ranking model in comparison with PageRank.

CHAPTER IV

CREDIBILITY-BASED LINK ANALYSIS

Most of the popular link-based Web ranking algorithms, like PageRank [138], HITS [100], and TrustRank [80], all rely on a fundamental assumption that the quality of a page and the quality of a page's links are strongly correlated: a page ranked higher will be unlikely to contain lower quality links. This assumption, however, also opens doors for spammers to create link-based Web spam that manipulate links to the advantage of the Web spammers.

Let us revisit two of the link-spam scenarios described in the previous section:

• **Hijacking:** Spammers hijack legitimate reputable pages and insert links that point to a spammer-controlled page, so that it appears to link analysis algorithms that the reputable page endorses the spam page.

• **Honeypots:** Instead of directly hijacking a link from a reputable page and risking exposure, spammers often create legitimate-appearing Web sites (*honeypots*) to induce reputable pages to voluntarily link to these spammer-controlled pages. A honeypot can then pass along its accumulated authority by linking to a spam page.

Both scenarios show how spammers can take advantage of the tight quality-credibility coupling to subvert popular link-based Web ranking algorithms and why the assumption that the quality of a page and the quality of a page's links are highly correlated is vulnerable to link-based Web spam.

In this chapter we advocate a clean separation of page quality and link (or reference) quality and argue that the intrinsic quality of a page should be distinguished from its intrinsic link credibility. Our goal is to assign each page a link credibility score defined in terms of link quality, not in terms of page quality. Such an approach is complementary to the source-centric analysis discussed in Chapter 3; it can be used in conjunction with either source-centric or page-based link approaches.

To guide our understanding of this problem, we address a number of important research

questions.

• Can we formally define the concept of credibility to provide a degree of separation between page quality and link quality?

• What are the factors impacting the computation of credibility, and to what degree do these factors impact the application semantics of credibility-based link analysis?

• Can and how will credibility be impacted by local versus global linkage information?

This chapter addresses each of these questions in detail to provide an in-depth understanding of link credibility. We develop a CredibleRank algorithm that incorporates credibility into an enhanced spam-resilient Web ranking algorithm. Concretely, we make three unique contributions: First, we introduce the concept of link credibility, identify the conflation of page quality and link credibility in popular link-based algorithms, and discuss how to decouple link credibility from page quality. Second, we develop several techniques for semi-automatically assessing link credibility for all Web pages, since manually determining the credibility of every page on the Web is infeasible. Another unique property of our link credibility assignment algorithms is to allow users with different risk tolerance levels to assess credibility in a personalized manner. Third, we present a novel credibility-based Web ranking algorithm - CredibleRank - which incorporates credibility information directly into the quality assessment of each page on the Web.

In addition, we develop a set of metrics for measuring the spam resilience properties of ranking algorithms and show how the credibility information derived from a small set of known spam pages can be used to support high accuracy identification of new (previously unknown) spam pages. We have conducted an extensive experimental study on the spam resilience of credibility-based link analysis over a Web dataset of over 100 million pages, and we find that our approach is significantly and consistently more spam-resilient than both PageRank and TrustRank.

In the next section (Section 4.1), we present the Web graph model and discuss three representative link-based ranking algorithms. We formally define credibility in Section 4.2, discuss several approaches for computing credibility in Section 4.3, and present a time-based

extension in Section 4.4. We show how to incorporate credibility information into an augmented Web ranking algorithm in Section 4.5. An extension for automatically expanding the scope of credibility information is introduced in Section 4.6. Finally, we report extensive experimental validation in Section 4.7, discuss some related work in Section 4.8, and summarize the contributions of this chapter in Section 4.9.

4.1 Reference Model

In this section, we present the Web graph model and discuss several popular approaches for link-based Web ranking. Readers familiar with these concepts can look forward to the following sections where we provide a formal definition of credibility and begin our study of relevant critical issues to computing and using credibility.

4.1.1 Web Graph Model

Let $\mathcal{G} = \langle \mathcal{P}, \mathcal{L} \rangle$ denote a graph model of the Web, where the vertexes in $\mathcal{P}$ correspond to Web pages and the directed edges in $\mathcal{L}$ correspond to hyperlinks between pages. For convenience, we assume that there are a total of n pages ($|\mathcal{P}| = n$) and that pages are indexed from 1 to n. A page $p \in \mathcal{P}$ sometimes is referred to by its index number i. A hyperlink from page p to page q is denoted as the directed edge $(p, q) \in \mathcal{L}$, where $p, q \in \mathcal{P}$. We denote the set of pages that p points to as $Out(p)$, and the set of pages that point to p as $In(p)$. Typically, each edge $(p, q) \in \mathcal{L}$ is assigned a numerical weight $w(p, q) > 0$ to indicate the strength of the association from one page to the other, where $\sum_{q \in Out(p)} w(p, q) = 1$. A common approach assigns each edge an equal weight (i.e. $w(p, q) = \frac{1}{|Out(p)|}$). Other approaches that favor certain edges are possible.

A Web graph $\mathcal{G}$ can be represented by an $n \times n$ transition matrix $\mathbf{M}$ where the ij^{th} entry indicates the edge strength for an edge from page i to page j. The absence of an edge from one page to another is indicated by an entry of 0 in the transition matrix:

$$M_{ij} = \begin{cases} w(i, j) & \text{if } (i, j) \in \mathcal{L} \\ 0 & \text{otherwise} \end{cases}$$

4.1.2 Link-Based Ranking: Overview

A number of link-based ranking algorithms have been proposed over the Web graph, including the popular HITS [100], PageRank [138], and TrustRank [80]. Most of these algorithms assume that a link from one page to another is counted as a "recommendation vote" by the originating page for the target page. To illustrate the core of link-based rank analysis, we below outline HITS, PageRank, and TrustRank.

HITS: The HITS algorithm uses a query dependent approach to ranking pages. Given a query result set, one can construct its neighbor graph, which is a subset of the Web graph. The HITS algorithm then assigns each page within the subset two rank scores: an *authority* score and a *hub* score. Typically the subset is related to a particular topic or query. The authority score indicates that the page is authoritative for the particular topic. The hub score indicates that the page points to many authoritative pages on that topic.

For m pages in the topic-specific Web subset we can denote the authority scores as the vector $\mathbf{a} = (a_1, a_2, ..., a_m)$ and the hub scores as the vector $\mathbf{h} = (h_1, h_2, ..., h_m)$. Suppose $\mathbf{U}$ denotes the transition matrix associated with just the pages belonging to this Web subset. We can write the HITS equations as follows:

$$\mathbf{a} = \mathbf{U}^T\mathbf{h} \text{ and } \mathbf{h} = \mathbf{U}\mathbf{a}$$

After sufficient iterations, it has been shown that the solution authority vector $\mathbf{a}$ converges to the principal eigenvector of $\mathbf{U}^T\mathbf{U}$, and the solution hub vector $\mathbf{h}$ converges to the principal eigenvector of $\mathbf{U}\mathbf{U}^\mathbf{T}$ [100].

PageRank: PageRank provides a single global authority score to each page on the Web based on the linkage structure of the entire Web. PageRank assesses the importance of a page by recursively considering the authority of the pages that point to it via hyperlinks. This formulation counts both the number of pages linking to a target page *and* the relative quality of each pointing page for determining the overall importance of the target page.

For n Web pages we can denote the PageRank authority scores as the vector $\mathbf{r_p} = (r_{p1}, r_{p2}, ..., r_{pn})$. The PageRank calculation considers the transition matrix $\mathbf{M}$ as well

as an n-length static score vector $\mathbf{e}$, which is typically taken to be the uniform vector $\mathbf{e} = \left(\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n}\right)$. We can write the PageRank equation as a combination of these two factors according to a mixing parameter α:

$$\mathbf{r_p} = \alpha \mathbf{M}^T \mathbf{r_p} + (1 - \alpha)\mathbf{e} \tag{7}$$

which can be solved using a stationary iterative method like Jacobi iterations [68]. To ensure convergence, pages with no outlinks are modified to include virtual links to all other pages in the Web graph (forcing $\mathbf{M}$ to be row stochastic).

TrustRank: PageRank's static score vector can also be used to bias the calculation toward certain pages based on exogenous (non-linkage) information. This bias was originally suggested in the original PageRank paper [138], and more fully explored in the context of personalizing PageRank in [83], where a non-uniform topic-sensitive static score vector was used to skew the PageRank scores toward pages relevant to a particular subject (e.g., sports). The recently proposed TrustRank algorithm [80] suggests biasing the PageRank calculation toward pre-trusted pages in an effort to suppress Web spam (similar to PageRank personalization suggested in [138] and more fully explored in [83]). Instead of considering the uniform static score vector $\mathbf{e}$, the TrustRank algorithm considers an n-length vector $\mathbf{v}$ that reflects a priori trust in each page. For example, in a Web graph of 5 pages, if pages 1 and 3 are pre-trusted then $\mathbf{v}$ could be set to $\mathbf{v} = \left(\frac{1}{2}, 0, \frac{1}{2}, 0, 0\right)$. For n Web pages the TrustRank scores can be denoted by the vector $\mathbf{r_t} = (r_{t1}, r_{t2}, ..., r_{tn})$, and we can write the TrustRank equation as:

$$\mathbf{r_t} = \alpha \mathbf{M}^T \mathbf{r_t} + (1 - \alpha)\mathbf{v} \tag{8}$$

Determining the a priori trust vector $\mathbf{v}$ is of critical importance, and a number of techniques have been suggested, including the use of expert-selected whitelists, high PageRank pages, and topically-segmented trusted pages [80, 180].

In summary, link-based ranking algorithms like HITS, PageRank, and TrustRank attempt to estimate a page's intrinsic quality by analyzing the hyperlink structure of the

Web. Fundamentally, the quality of a page and the quality of its links are tightly coupled in each of these ranking algorithms. Returning to Equation 7, a page with a PageRank score of r contributes $\alpha \cdot r$ to the pages that it links to, where α is typically in the range $[0.75, 0.95]$. We have discussed that such tight coupling is not only inadequate in practice but also creates Web spam vulnerabilities. In the rest of this chapter we present the concept and the usage of credibility-based link analysis.

4.2 Link Credibility

In this section, we formally introduce the concept of credibility in terms of *k-Scoped Credibility*. We shall ground our discussion of link credibility in the context of Web spam and explore how to algorithmically determine a page's credibility. Concretely, let C be a *credibility function* that instantaneously evaluates the link quality of a Web page p at time t. A score of $C(p,t) = 0$ indicates that the page p is not credible in terms of its links at time t. In contrast, a score of $C(p,t) = 1$ indicates that the page p is perfectly credible in terms of its links at time t. We observe that a desirable credibility function should have the following qualities:

• First, we observe that a page's link quality should depend on its own links and perhaps is related to the link quality of its neighbors up to some small number (k) of hops away. Hence, link credibility of pages should be a function of the local characteristics of a page and its place in the Web graph, and not the global properties of the entire Web (as in a PageRank-style approach).

• Second, we observe that relying heavily on a set of known good pages (a whitelist) may be problematic. Spammers may attempt to mask their low quality links to spam pages by linking to known whitelist pages. Also, relying too heavily on a whitelist for link credibility assignment makes these pages extremely valuable for spammers to corrupt.

• Third, we observe that a page's credibility should be related to its distance to known spam pages (a blacklist) to penalize pages for poor quality links. Since blacklist pages may be abandoned by spammers once they are identified, a credibility function should be robust to such transience.

Generally speaking, the Web is too large and too quickly growing to manually label each page as either spam or not spam. We shall assume that the set $\mathcal{P}$ of all pages can be divided into the set of known good pages, denoted by $\mathcal{P}_w$ (the whitelist), the set of known spam pages, denoted by $\mathcal{P}_b$ (the blacklist), and the set of pages for which the user has no experience or judgment, denoted by $\mathcal{P}_u$ (the unknown pages), such that $\mathcal{P} = \mathcal{P}_w \cup \mathcal{P}_b \cup \mathcal{P}_u$. In practice, only a fraction of all pages on the Web will belong to either the whitelist or the blacklist($|\mathcal{P}_w|, |\mathcal{P}_b| << |\mathcal{P}|$).

4.2.1 Naive Credibility

We begin our analysis of credibility functions by considering a simple approach that illustrates some of our observations above and serves as a comparison to the k-Scoped Credibility function. The naive credibility function assigns a whitelist page a perfect credibility score of value one, a blacklist page no credibility (value zero), and an unknown page a default credibility value θ, $(0 < \theta < 1)$:

$$C_{naive}(p,t) = \begin{cases} 0 & \text{if } p \in P_b \\ \theta & \text{if } p \in P_u \\ 1 & \text{if } p \in P_w \end{cases}$$

The advantage of this naive credibility assignment is its ease of evaluation. However it has several apparent drawbacks – (i) it makes no effort to evaluate credibility in terms of the links of a page; (ii) the majority of all pages (P_u) receive a default credibility value; and (iii) whitelist pages, though generally high-quality, may not necessarily be perfectly credible in reality at all times.

4.2.2 k-Scoped Credibility

We next introduce k-Scoped Credibility, which evaluates the credibility of a page in terms of the quality of a random walk originating from the page and lasting for up to k steps. Critical to this *k-Scoped Credibility* is the notion of a *path*.

Definition 1 (path) Consider a directed Web graph $\mathcal{G} = \langle \mathcal{P}, \mathcal{L} \rangle$, an originating page p and a destination page q. A *path* in the directed graph $\mathcal{G}$ from page p to page q is a sequence

of nodes: $path(p,q) = \langle n_0, n_1, ..., n_j \rangle$ (where $p = n_0$ and $q = n_j$) such that there exists a directed edge between successive nodes in the path, $(n_i, n_{i+1}) \in \mathcal{L}$ for $0 \leq i \leq j-1$. The length $|path(p,q)|$ of a path is j, the number of edges in the path. There may exist multiple paths from p to q.

We refer to the set of all paths of a specified length (say, k) that originate from a page p as $Path_k(p)$. We will sometimes refer to a specific path of length k originating from p using the notation $path_k(p)$, where $path_k(p) \in Path_k(p)$.

Our notion of k-Scoped Credibility relies on a special type of path that we call a *bad path*.

Definition 2 (bad path) Consider a directed Web graph $\mathcal{G} = \langle \mathcal{P}, \mathcal{L} \rangle$, an originating page p and a destination page q. We say that a path in the directed graph $\mathcal{G}$ from page p to page q is a *bad path* if the destination page is a spam page, $q \in \mathcal{P}_b$, and no other page in the path is a spam page. $path(p,q) = \langle n_0, n_1, ..., n_j \rangle$ (where $p = n_0$ and $q = n_j$) and $q \in \mathcal{P}_b$ and $n_i \notin \mathcal{P}_b$, for $0 \leq i \leq j-1$.

We refer to the set of all bad paths of a specified length (say, k) that originate from a page p as $BPath_k(p)$.

The probability of a random walker travelling along a k-length path originating at page p is denoted by $Pr(path_k(p))$, and is determined by the probabilistic edge weights for each hop of the path:

$$Pr(path_k(p)) = \prod_{i=0}^{k-1} w(n_i, n_{i+1})$$

Formally, we define the *k-Scoped Credibility* of a page in terms of the probability that a random walker *avoids* spam pages after walking up to k hops away from the originating page. For $k = 1$, the k-Scoped Credibility is simply the fraction of a page's links that point to non-spam pages. Increasing k extends the scope of this credibility function by considering random walks of increasing length. For an originating page $p \in \mathcal{P}$, if p is a spam page, we set its link credibility to be 0, regardless of the characteristics of the pages it links to.

Definition 3 (k-Scoped Credibility) Let $\mathcal{G} = \langle \mathcal{P}, \mathcal{L} \rangle$ be a directed Web graph, k be a maximum walk radius where $k > 0$, and $p \in \mathcal{P}$ be a page in the Web graph. The *k-Scoped Credibility* of page p at time t, denoted by $C_k(p,t)$, is defined as follows:

$$C_k(p,t) = 1 - \sum_{l=1}^{k} \left(\sum_{path_l(p) \in BPath_l(p)} Pr(path_l(p)) \right)$$

For the special case when $p \in \mathcal{P}_b$, let $C_k(p,t) = 0$.

In the case that there are no spam pages within k hops of page p, then p is perfectly credible: $C_k(p,t) = 1$. In the case that p itself is a spam page or in the case that all paths originating at page p hit a spam page within k hops, then p is not credible at all: $C_k(p,t) = 0$. Intuitively, the k-Scoped Credibility function models a random walker who when arriving at a spam page, becomes stuck and ceases his random walk, and for all other pages the walker continues to walk, for up to k hops.

4.3 Computing Credibility

In practice, of course, the k-Scoped Credibility function can only have access to some portion of the entire Web graph, due to the size of the Web, its evolution, and the cost of crawling all pages. Additionally, only some spam pages will be known to the credibility function through the blacklist. In order to correct the inaccuracy in computing k-Scoped Credibility due to the presence of an incomplete Web graph and a partial blacklist, in this section we introduce the concept of tunable k-Scoped Credibility, which augments the basic k-Scoped Credibility computation by including a credibility penalty factor as a control knob. Our goals are to better approximate the k-Scoped Credibility under realistic constraints and understand how different parameters may influence the quality of a credibility function.

4.3.1 Tunable k-Scoped Credibility

The tunable k-Scoped Credibility is a function of two components: a random-walk component with respect to the known bad paths (based on the blacklist) and a penalty component. The penalty component is intended to compensate for the bad paths that are unknown to the credibility function. We first define the tunable k-Scoped Credibility and then focus our

discussion on alternative approaches for assigning the credibility discount factor to offset the problem of an incomplete Web graph and a partial blacklist.

Definition 4 (Tunable k-Scoped Credibility) Let $\mathcal{G} = \langle \mathcal{P}, \mathcal{L} \rangle$ be a directed Web graph, k be a maximum walk radius where $k > 0$, and $\gamma(p)$ be the credibility penalty factor of a page $p \in \mathcal{P}$ where $0 \leq \gamma(p) \leq 1$. We define the *tunable k-Scoped Credibility* of page p, denoted by $C_k(p)$, in two steps: when $p \notin \mathcal{P}_b$:

$$C_k(p) = \left(1 - \sum_{l=1}^{k} \left(\sum_{path_l(p) \in BPath_l(p)} Pr(path_l(p)) \right) \right) \cdot \gamma(p)$$

In the case of $p \in \mathcal{P}_b$, let $C_k(p) = 0$.

The penalty factor $\gamma(p)$ is an important tunable parameter of the credibility assignment and can be used as the credibility discount knob. Since the blacklist $\mathcal{P}_b$ provides only a partial list of all spam pages in the Web graph at a given point of time, the penalty factor can be used to update the random walk portion of the credibility calculation to best reflect the possible spam pages that are not yet on the blacklist. We rely on a hop-based approach for determining the proper credibility discount factor in computing k-Scoped Credibility for each page in the Web graph. To better understand the advantage of our hop-based approach, we also discuss the optimistic and pessimistic approaches as two extremes for selecting the penalty factor for each page.

4.3.1.1 The Optimistic Approach

This approach defines the credibility penalty factor for a page by assigning no penalty at all. In other words, for all pages, we assign a credibility discount factor of 1:

$$\gamma_{opt}(p) = 1, \forall p \in \mathcal{P}$$

meaning the random walk component of the tunable k-Scoped Credibility is not penalized at all. We call this approach an *optimistic* one since it is equivalent to assuming that all spam pages are on the blacklist. The optimistic approach will tend to over-estimate the credibility of pages that link to the spam pages not on the blacklist.

4.3.1.2 The Pessimistic Approach

In contrast, a *pessimistic* approach treats a page with *any* j-length path ($1 \leq j \leq k$) leading to a blacklist page as *not credible* within the k-hop scope in the sense that *all* paths originating from such a page are considered bad paths.

$$\gamma_{pess}(p) = \begin{cases} 0 & \text{if } |BPath_j(p)| \geq 1 \text{ for any j, } 0 < j \leq k \\ 1 & \text{otherwise} \end{cases}$$

A pessimistic approach may be appropriate in circumstances when links to spam pages are highly correlated (e.g., if the presence of a link to a blacklist page is always accompanied by another link to a spam page that is not on the blacklist). In many circumstances, the presence of a single bad path originating at a page may be the result of a temporary linking mistake (as in the hijacking example discussed in the introduction) or truly indicative that the page has only a fraction of links leading to spam pages. Hence, the pessimistic approach may be too draconian.

4.3.1.3 The Hop-Based Approach

The third approach for determining the credibility discount factor balances the extremes of the optimistic and pessimistic approaches by considering the number and the length of the bad paths for a page. A bad path is treated as evidence that there are other bad paths for an originating page that have been overlooked due to the partial nature of the blacklist and the incompleteness of the Web graph.

For a bad path of length j originating at page p, we associate a hop-based discount factor $\gamma_{hop,j}(p)$, where $0 \leq \gamma_{hop,j}(p) \leq 1$. By default, we let $\gamma_{hop,j}(p) = 1$ if there are no bad paths of length j originating from p (i.e. $BPath_j(p) = \emptyset$). The hop-based discount factor can then be calculated as the product of the constituent discount factors: $\gamma_{hop}(p) = \prod_{j=1}^{k} \gamma_{hop,j}(p)$.

Determining the choice of hop-based discount factors is critical to the quality of tunable k-scoped credibility calculation. We below discuss three alternative ways to compute $\gamma_{hop,j}(p)$. We begin with a user-defined discount factor ψ ($0 < \psi < 1$) to set the initial hop-based discount for bad paths of length 1, i.e., $\gamma_{hop,1}(p) = \psi$. Then we introduce three

approaches for damping the user-defined discount factor that determine how quickly the discount factor approaches 1 as path length increases. Setting ψ close to 0 will result in a more pessimistic credibility penalty, whereas ψ close to 1 is intuitively more optimistic. By tuning ψ and the damping function we can balance these extremes.

Constant: One way for damping the initial setting of the user-defined discount factor ψ for bad paths of increasing length is to penalize all paths of varying lengths emanating from a page equally if there exists one bad path, i.e., $BPath_j(p) \neq \emptyset$. We refer to this approach as a *constant* discount factor since the hop-based discount does not vary with bad path length:

$$\gamma_{hop,i}(p) = \psi$$

Using a constant damping factor, a page that directly links to a spam page results in a credibility penalty that is the same as the penalty for a more distant path to a spam page.

Linear: The second approach to set the discount factor is *linear* in the length of a bad path up to some pre-specified path length L. Paths of distance L or greater are considered too distant to provide additional evidence of other bad paths, and so the discount factor is 1 for those paths.

$$\gamma_{hop,i}(p) = \begin{cases} \dfrac{(i-1)}{L-1}(1-\psi) + \psi & \text{if } i < L \\ 1 & \text{otherwise} \end{cases}$$

Using a linear damping factor, a path to a spam page that is farther away from the originating page results in a less severe hop-based discount than the credibility penalty for a direct link or short path from the originating page p to a spam page.

Exponential: The third approach for setting the discount factor is *exponential* in the length of the path, meaning that the initial discount factor ψ for bad paths of length 1 is quickly damped close to 1 as the bad path length increases.

$$\gamma_{hop,i}(p) = 1 - (1-\psi)\psi^{i-1}$$

Compared with the linear damping factor, the exponential damping factor allows the

credibility discount for a spam page to be quickly damped close to 1. Put differently, when a spam page is closer to the originating page p, the link credibility of p is discounted less than the linear case with respect to the hop count.

4.3.2 Implementation Strategy

The k-Scoped Credibility is a local computation, requiring only an originating page and a forward crawl of all pages within k hops of the originating page. Hence, the credibility of a page can be updated in a straightforward fashion and as often as the k-hop neighbors are refreshed via Web crawls.

In practice, we anticipate computing the k-Scoped Credibility in batch for all Web pages in the current Web graph state after each Web crawl. The main cost of computing the tunable k-Scoped Credibility is the cost of identifying the set of bad paths for each page and the cost of explicitly computing the path probabilities (recall Section 4.2.2). We calculate the tunable k-Scoped Credibility for all pages using an equivalent iterative approach that is cheaper and faster. Let $\mathcal{G} = \langle \mathcal{P}, \mathcal{L} \rangle$ denote a graph model of the Web and $|\mathcal{P}| = n$ (recall Section 4.1.1). We first construct an n-length indicator vector $\mathbf{d} = (d_1, d_2, \ldots, d_n)$ to reflect whether a page is in the blacklist or not:

$$d_i = \begin{cases} 1 & \text{if } p_i \in P_b \\ 0 & \text{otherwise} \end{cases}$$

We next construct an $n \times n$ transition matrix $\mathbf{B}$ that replicates the original Web graph transition matrix $\mathbf{M}$, but with transition probabilities exiting a blacklist page of 0, to indicate the random walker stops when he arrives at a blacklist page.

$$B_{ij} = \begin{cases} M_{ij} & \text{if } d_i \notin \mathcal{P}_b \\ 0 & \text{otherwise} \end{cases}$$

In practice, the matrix $\mathbf{B}$ need not be explicitly created. Rather, the original matrix $\mathbf{M}$ can be augmented with rules to disregard blacklist entries.

The penalty factors can be encoded in an $n \times n$ diagonal matrix $\mathbf{\Gamma}$, where the diagonal elements correspond to the per-page penalty factors:

$$\Gamma_{ij} = \begin{cases} \gamma(i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Finally, the n-length tunable k-Scoped Credibility vector $\mathbf{C}_{[k]}$ can be computed:

$$\mathbf{C}_{[k]} = \left(\mathbf{1} - \sum_{j=0}^{k} \mathbf{B}^{(j)} \mathbf{d}^T \right) \cdot \mathbf{\Gamma}$$

where $\mathbf{1}$ is an n-length vector of numerical value 1s. Note that the matrix multiplication of $\mathbf{\Gamma}$ can be implemented as an element-wise vector product, so the expense of a matrix-by-matrix multiplication can be largely avoided.

4.4 Time-Sensitive Credibility

Recall that the k-Scoped Credibility function described in Section 4.2 can be instantaneously evaluated for a page over the entire Web graph. To simplify the presentation, in all of the tunable credibility functions described so far, we have considered a single (incomplete) Web graph snapshot for assessing link credibility. Since the Web is constantly evolving and spammers are known to shift tactics, the credibility functions defined based on one snapshot of the Web graph may not accurately reflect the actual credibility of each page at any given time t. In this section, we extend these credibility functions to consider a history of Web graph snapshots in an effort to reflect the actual k-Scoped Credibility of each page in the presence of an evolving Web graph.

4.4.1 Credibility Over Time

We maintain multiple snapshots of the Web graph as it evolves over time. Since resources are limited for collecting snapshots (via Web crawls), we consider a finite sequence of snapshots $\mathcal{E}$ consisting of a current snapshot, plus up to H additional history snapshots. For presentation clarity, we assume that the snapshots are collected at regular intervals, but our approach in this section can be easily extended to consider non-regular intervals.

$$\mathcal{E} = \langle \mathcal{G}_{[t-H]}, ..., \mathcal{G}_{[t-2]}, \mathcal{G}_{[t-1]}, \mathcal{G}_{[t]} \rangle$$

Our goal is to assign each page in the current Web graph snapshot a time-sensitive credibility score over its entire link history using a weighted average over a sequence of snapshot-based credibility calculations. By considering a history of a page's links, we are able to assess the link credibility of the page and assign a better quality credibility score by incorporating more evidence on how the quality of the links in the page evolves over time. We may discover bad paths in some snapshots that are not present in the current snapshot merely due to the incomplete nature of the current Web crawl. Similarly, we may have a page that historically has had many bad paths (and hence, low credibility), but then has no bad paths in our most recent crawl.

We compute the time-sensitive credibility score for a page p over a Web graph history $\mathcal{E}$ as follows:

$$C_{[\mathcal{E}]}(p) = \sum_{i=0}^{H} C_{[t-i]}(p) \cdot \frac{I_{[t-i]}}{\sum_{j=0}^{H} I_{[t-j]}}$$

where $C_{[t]}(p)$ is a credibility function (like the ones introduced in the previous sections) evaluated over $\mathcal{G}_{[t]}$; and $I_{[t]}$ is an importance weight associated with a page's credibility for the Web graph snapshot at time t.

There are two challenges for implementing the time-sensitive weighted average credibility function – the choice of importance weight and the policy for assigning credibility scores to pages that are not present in all snapshots.

4.4.2 Choice of Importance Weight

One choice of importance weight is the exponentially weighted sum: $I_{[t-i]} = \lambda^i$ for $0 \leq i \leq H$. For $\lambda < 1$, the time-averaged credibility favors more recent credibility scores for a page over older credibility scores. Letting $\lambda = 1$ results in a simple average of the credibility scores for the history. Alternatively, we could treat evidence of bad paths at any point in a page's link history as a strong predictor for a page's current credibility, regardless of the page's most recent link state. For such a scenario, we can select weights that favor snapshots in which a page's credibility is low.

4.4.3 Missing Page Policy

Since a page may be present in some Web snapshots but not all (because it is a newly created page or missing from some Web crawls), the time-averaged credibility function needs a policy for handling missing pages. We consider two approaches for assigning a page not present in the $G_{[t-j]}$ snapshot a default credibility score $C_{[t-j]}(p)$. The first assigns the page the best credibility score from across the entire Web graph history: $C_{[t-j]}(p) = max_{i=0}^{H} C_{[t-i]}(p)$. The second assigns the page the worst credibility score from across the entire Web graph history: $C_{[t-j]}(p) = min_{i=0}^{H} C_{[t-i]}(p)$. There are other alternatives, including assigning a missing page a default value or the average credibility value over the existing snapshots. We study the *max* and *min* policies only in our experiments.

4.5 Credibility-Based Web Ranking

We have presented the design of several credibility functions for evaluating Web page link quality. In this section, we use this decoupled credibility information to augment the page quality assessment of each page on the Web with a goal of suppressing Web spam. Concretely, we demonstrate how link credibility information can improve PageRank and TrustRank-style approaches through a credibility-based Web ranking algorithm called CredibleRank.

Returning to PageRank (see Equation 7), there are several avenues for incorporating link credibility information. We outline four alternatives below:

• First, the initial score distribution for the iterative PageRank calculation (which is typically taken to be a uniform distribution) can be seeded to favor high credibility pages. While this modification may impact the convergence rate of PageRank, it has no impact on ranking quality since the iterative calculation will converge to a single final PageRank vector regardless of the initial score distribution.

• Second, the graph structure underlying the transition matrix $\mathbf{M}$ can be modified to remove low credibility pages and edges to low credibility pages. While this modification may eliminate some Web spam pages, it could also have the negative consequence of eliminating legitimate pages that are merely of low credibility.

• Third, the edge weights in the transition matrix $\mathbf{M}$ can be adjusted to favor certain edges, say edges to high-credibility pages. While this change may have some benefit, a low credibility page p's overall influence will be unaffected (since $\sum_{q \in Out(p)} w(p,q) = 1$).

• Finally, the static score vector $\mathbf{e}$ can be changed to reflect the link credibility information, much like in TrustRank and Personalized PageRank [80, 83]. By skewing $\mathbf{e}$ toward high credibility pages (or away from low credibility pages) we can give a ranking boost to these pages, which could have the undesired consequence of ranking a low-quality high-credibility page over a high-quality low-credibility page.

Alternatively, we advocate a credibility-augmented Web ranking algorithm that uses credibility information to impact the size of the vote of each page. CredibleRank asserts that a page's quality be determined by two criteria: (1) the quality of the pages pointing to it; and (2) the credibility of each pointing page. A link from a high-quality/high-credibility page counts more than a link from a high-quality/low-credibility page. Similarly, a link from a low-quality/high-credibility page counts more than a link from a low-quality/low-credibility page. By decoupling link credibility from the page's quality, we can determine the credibility-augmented quality of each page through a recursive formulation.

Recall that $In(p)$ denotes the set of pages linking to p. We compute the CredibleRank score $r_c(p)$ for a page p as:

$$r_c(p) = \sum_{q \in In(p)} C(q) \cdot r_c(q) \cdot w(q,p)$$

This formula states that the CredibleRank score (quality) of page p is determined by the quality ($r_c(q)$) and the link credibility ($C(q)$) of the pages that point to it, as well as the strength of the link $w(q,p)$. In this sense, the link weights are used to determine how a page's "vote" is split among the pages that it points to, but the credibility of a page impacts how large or small is the page's vote.

We can extend this formulation to consider n Web pages, where we denote the CredibleRank authority scores by the vector $\mathbf{r_c} = (r_{c1}, r_{c2}, ..., r_{cn})$. Recall that $\mathbf{M}$ denotes the $n \times n$ Web transition matrix, and $\mathbf{v}$ is an n-length static score vector. We can construct an $n \times n$ diagonal credibility matrix $\mathbf{CR}$ from the link credibility vector $\boldsymbol{\gamma}$, where the elements

of the credibility matrix are defined as:

$$CR_{ij} = \begin{cases} C(i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

We can then write the CredibleRank vector $\mathbf{r_c}$ as:

$$\mathbf{r_c} = \alpha(\mathbf{CR} \cdot \mathbf{M})^T \mathbf{r_c} + (1 - \alpha)\mathbf{v} \tag{9}$$

which, like PageRank and TrustRank, can be solved using a stationary iterative method like Jacobi iterations. The matrix multiplication of **CR** and **M** can be implemented as an element-wise vector product to avoid the expense of a matrix-by-matrix multiplication.

4.6 Blacklist Expansion

We have presented our mechanisms to compute the time-sensitive link credibility for pages in a Web graph and described the CredibleRank algorithm for spam-resilient Web page ranking, all based on the availability of a small blacklist $\mathcal{P}_b$. In this section we discuss our strategy for semi-automatically extending the small manually-created blacklist. Maintaining a high-quality up-to-date blacklist is an important and challenging problem, not only because our credibility functions are dependent on blacklist quality, but also because pages that are identified as spam are often abandoned by Web masters once discovered, and spammers continue to construct and push new spam pages to the Web over time. In this section, we briefly describe our development of a semi-supervised blacklist expansion algorithm using link and credibility information alone.

We have assumed the existence of an initial blacklist $\mathcal{P}_b$, where the pages on the blacklist may have been identified by human experts, trusted authorities (like SiteAdvisor), or algorithmic approaches, e.g. [62, 77, 134]. We are interested in using these blacklist pages to identify unknown spam pages from the space of all other Web pages ($\mathcal{P} - \mathcal{P}_b$) for addition to the blacklist. While a number of factors can be used to support blacklist expansion, we focus here on using link and credibility information alone.

While a number of popular classifiers are available (e.g., naive bayes, SVM), we have

adopted a threshold-based inverse PageRank approach to assign a *spam-likelihood* value to all pages in the Web graph. To derive spam-likelihood for all pages, we propagate a spam-likelihood score to every page in the Web graph using an iterative process similar to the PageRank calculation in Equation 7. We denote the spam-likelihood score for page p as $s(p)$. The n-length spam-likelihood vector is denoted by $\mathbf{s}$. We say that $s(p) > s(q)$ means that page p's spam-likelihood is greater than page q.

We begin by reversing the links in the original page graph $\mathcal{G} = \langle \mathcal{P}, \mathcal{L} \rangle$ so that we have a new *reversed* page graph $\mathcal{G}' = \langle \mathcal{P}, \mathcal{L}' \rangle$, where every edge $(p, q) \in \mathcal{L} \Rightarrow (q, p) \in \mathcal{L}'$. Hence, a page that is pointed to by many other pages in the original graph will now itself point to those pages in the reversed graph. Corresponding to the reversed page graph is a transition matrix $\mathbf{M_r}$ (where, typically $\mathbf{M_r} \neq \mathbf{M}^T$). We then create a static score vector that is biased toward low credibility pages (like blacklist pages and those pages pointing to blacklist pages): $\mathbf{1} - \mathbf{C}_{[k]}$. We can then compute the spam likelihood vector $\mathbf{s}$ as the solution to:

$$\mathbf{s} = \beta {\mathbf{M_r}}^T \mathbf{s} + (1 - \beta)(\mathbf{1} - \mathbf{C}_{[k]})/||\mathbf{1} - \mathbf{C}_{[k]}||$$

where $||\mathbf{1} - \mathbf{C}_{[k]}||$ is the L1-norm and β is a mixing parameter. Spam-likelihood models a "reverse" random walker who visits the pages that point to a given page, and, when he gets bored, resets to low credibility pages. Intuitively, a page will receive a high spam-likelihood score if it points to pages with high spam-likelihood. Pages that are distant from spam pages will receive lower spam-likelihood scores. Note that unlike our credibility formulation, spam-likelihood leverages the global structure of the Web and provides a global rank to pages (and not a per-page probabilistic interpretation). In our experiments section, we study the accuracy of a threshold-based spam classifier that classifies as spam all pages with spam-likelihood above some threshold.

4.7 Experimental Evaluation

In this section, we report the results of an experimental study of credibility-based link analysis over a Web dataset of over 100 million pages. We report three sets of experiments – (1) an evaluation of tunable k-Scoped Credibility and the factors impacting it (like scope

k, discount factor, blacklist size, time-sensitivity, and damping function); (2) a study of the spam-resilience characteristics of CredibleRank, where we show that our approach is significantly and consistently more spam-resilient than both PageRank and TrustRank; and (3) an evaluation of our credibility-based blacklist expansion.

4.7.1 Setup

The experiments reported in this chapter use the Stanford WebBase dataset (described in Chapter 4) consisting of 118 million pages and 993 million links. The dataset was originally collected in 2001 and includes pages from a wide variety of top-level-domains.

Defining what exactly constitutes *spam* is an open question, and so as a baseline for our experiments we considered pornography related pages in the dataset as spam. Naturally, this is one of many possible spam definitions and we anticipate revisiting this topic in our continuing research. Since manually inspecting all 118 million pages is an onerous task, we applied a simple procedure to identify spam pages. We first identified all sites with a URL containing a pornography-related keyword (where we define a site by the host-level information embedded in each page's URL). This resulted in 11,534 sites and over 1.5 million pages. For these 11,534 sites, we then sampled a handful of pages from each site and kept only those sites that we judged to be spam. Applying this filter yielded 9,034 sites consisting of 1,202,004 pages. We refer to these pages as the *Spam Corpus*.[1]

We generated three blacklists by randomly selecting sites from the Spam Corpus. The first blacklist (referred to as **Large**) contains 20% of the sites in the Spam Corpus (1807 sites, or 0.24% of all sites); the second blacklist (**Medium**) contains 10% of the Spam Corpus (903 sites); the third blacklist (**Small**) contains just 1% (90 sites). This spam set does not contain all pornography pages in the dataset, but we can be assured that all pages in it are indeed spam (i.e., there are no false positives).

For the whitelist, we manually selected 181 sites from the top-5000 sites (as ranked by PageRank). These whitelist sites are each maintained by a clearly legitimate real-world entity, either a major corporation, university, or organization. We additionally ensured that

[1] In a 1964 U.S. Supreme Court case, Justice Potter Stewart famously declared that obscenity was hard to define but that "I know it when I see it". We concur.

each of these 181 sites was not within two-hops of any site in the Spam Corpus.

We grouped pages in the entire dataset into sites (again, by the host information of each page's URL), resulting in 738,626 sites. We constructed a site graph where each site is a node in the graph. If a page in one site points to a page in another site we included an edge in the site graph, excluding self-edges. The result is 11,816,108 edges in the site-level Web graph. We adopted a fairly standard approach for defining the edge strength for a site-level edge as the fraction of page-level hyperlinks pointing from the originating site to the target site (e.g., [97, 173]), and constructed the transition matrix $\mathbf{M}$ based on these edge weights. For all ranking calculations, we relied on the standard mixing parameter $\alpha = 0.85$ used in the literature (e.g., [80, 138]), and we terminated the Jacobi method after 50 iterations.

Site-level link analysis has several appealing characteristics – iterative ranking algorithms are considerably faster than over page-level graphs and links internal to a site are discounted. Also, spam sites may include a mix of legitimate pages and spam pages; since all the pages are controlled by a spammer, it is reasonable to mark all pages as spam. The techniques presented here can be trivially applied to the page-level Web graph as well.

The data management component of the ranking algorithms was based on the WebGraph compression framework described in [26] for managing large Web graphs in memory.

4.7.2 Credibility Assignment Evaluation

In the first set of experiments, we evaluate tunable k-Scoped Credibility and the many factors impacting it.

4.7.2.1 Credibility Coverage

In our first experiment (shown in Table 7), we examine how widely the tunable k-Scoped Credibility can assign link credibility scores to sites beyond the pre-labelled blacklist. By increasing the tunable scope parameter k, the credibility function will consider paths of increasing length, meaning that there will be a greater opportunity for identifying bad paths. We measure the *coverage* of k-Scoped Credibility in terms of the scope parameter k, a complete blacklist b (the Spam Corpus), and a partial blacklist b', where $b' \subset b$:

Table 7: Credibility Coverage

	Blacklist Size		
Scope (k)	**Small**	**Medium**	**Large**
1	7%	27%	39%
2	5%	33%	46%
3	26%	73%	79%
4	75%	94%	95%
5	95%	98%	98%
10	99%	99%	99%

$$cov(k, b, b') = \frac{|\{p \in \mathcal{P} | \exists j, 1 \leq j \leq k \text{ s.t. } BPath_j(p, b') \neq \emptyset\}|}{|\{p \in \mathcal{P} | \exists j, 1 \leq j \leq k \text{ s.t. } BPath_j(p, b) \neq \emptyset\}|}$$

where $BPath_k(p, b)$ denotes the set of all bad paths to sites in blacklist b of length k that originate from a page p. The numerator corresponds to the count of all sites with at least one path to a site on the partial blacklist. The denominator corresponds to the count of all sites with at least one path to a site on the complete blacklist (the Spam Corpus). So, for $k = 1$, there are 18,305 sites that are either spam sites or directly link to spam sites; of these 27% are on the Medium blacklist or directly link to a site on the Medium blacklist.

There are three interesting observations. First, the size of the blacklist is important. A larger blacklist leads to more evidence of bad paths, and hence will give our tunable k-Scoped Credibility function the opportunity to make higher-quality credibility assessments, even for small k. Second, even for the Small blacklist – with just 90 sites – we find fairly reasonable coverage (26%) for $k = 3$, indicating that there is some linking correlation for pages in our dataset with respect to the Spam Corpus. While this correlation may not be present for all types of spam, it is encouraging and an avenue we anticipate studying more in our continuing research. Third, for large k, nearly all pages have at least one path to a spam page. Thus, pages that are quite distant from an originating page likely have little impact over the credibility of the originating page. The choice of k should be made with care, and so we must be careful in our application of the credibility penalty factor. As we will see in our next set of experiments, a Pessimistic penalty factor will result in nearly all sites being assigned very low credibility if a large scope is considered.

4.7.2.2 Credibility Quality

We next study the quality of the tunable k-Scoped Credibility function over different settings of the credibility penalty factor (Figures 33, 34, 35, 36, and 37). Recall that the penalty factor is used to update the random walk portion of the credibility calculation to reflect the possible spam pages (or sites, in this case) not yet on the blacklist. Our goal is to understand how well the tunable k-Scoped Credibility functions perform as compared to the k-Scoped Credibility with access to the full Spam Corpus.

We consider five different settings for the penalty factor of the k-Scoped Credibility – the Optimistic, Pessimistic, and three Hop-Based approaches. For each of the Hop-Based approaches – constant, linear, and exponential – we report the results for an initial credibility discount factor $\psi = 0.5$. For each of these 5 settings, we calculated the credibility for each site using only 10% of all spam sites (the Medium blacklist b').

We evaluate the error for each of these credibility functions over the Medium blacklist b' versus the actual credibility computed over the entire Spam Corpus b. We measure the overall error for a tunable credibility function C over b' as the average of the pair-wise credibility differences with the actual credibility function C^* over b:

$$error(C, b, b') = \frac{1}{|X|} \sum_{p \in X} |C^*(p) - C(p)|$$

where X is the set of sites with at least one bad path to a site in the Spam Corpus: $X = \{p \in \mathcal{P} | \exists j, 1 \leq j \leq k \text{ s.t. } BPath_j(p, b) \neq \emptyset\}$.

In Figure 33, we report the average credibility error for each of the five tunable k-Scoped Credibility functions evaluated over increasing values of the scope parameter k.

There are three interesting observations. First, the Optimistic penalty factor performs very well, resulting in the lowest average credibility error for $k >= 2$. This indicates that the credibility scores assigned by the Optimistic approach are the closest to the scores assigned by the credibility function with access to the entire Spam Corpus. The Optimistic approach assigns nearly all sites a credibility close to 1; on inspection, we discover that the actual credibility of most sites is close to 1, and so the optimistic approach does well by defaulting

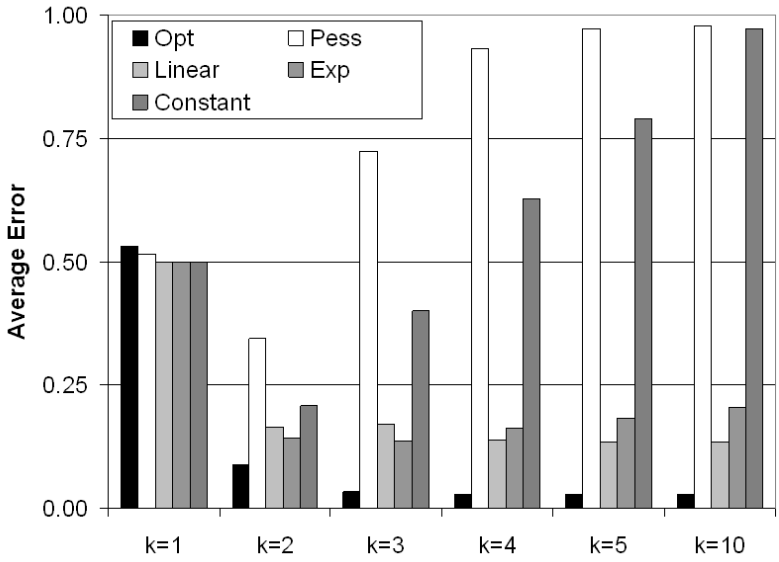

Figure 33: Average Credibility Error - Varying k

to a high credibility score for most sites.

Second, the Pessimistic and Constant penalty factors perform well for $k = 2$, and then increasingly worse as the scope parameter k increases. These two approaches are very pessimistic, assigning 0 or low credibility to sites with even a single bad path. For $k = 2$, only sites within a close radius of sites on the blacklist are penalized. Thus we see a fairly low error rate. As k increases, most sites have at least one path to a blacklist site (recall Table 7), and are assigned a 0 or low credibility score, resulting in a high error rate.

Third, the Exponential and Linear approaches result in better performance than Pessimistic and Constant, but worse than Optimistic. As k increases, the error increase observed in the Constant approach is avoided since the Exponential and Linear penalty factors treat long paths less severely. On further inspection, we discovered that only these two approaches balance the credibility over-estimation of the Optimistic approach and the under-estimation of the Pessimistic and Constant approaches.

To further illustrate this over and under estimation balance, we next report the distribution of credibility scores based on the Medium blacklist for the Optimistic, Pessimistic, and Hop-Based (exponential) approaches for all sites that point to sites in the Spam Corpus. Figures 34, 35, and 36 report the distribution of credibility scores versus the actual

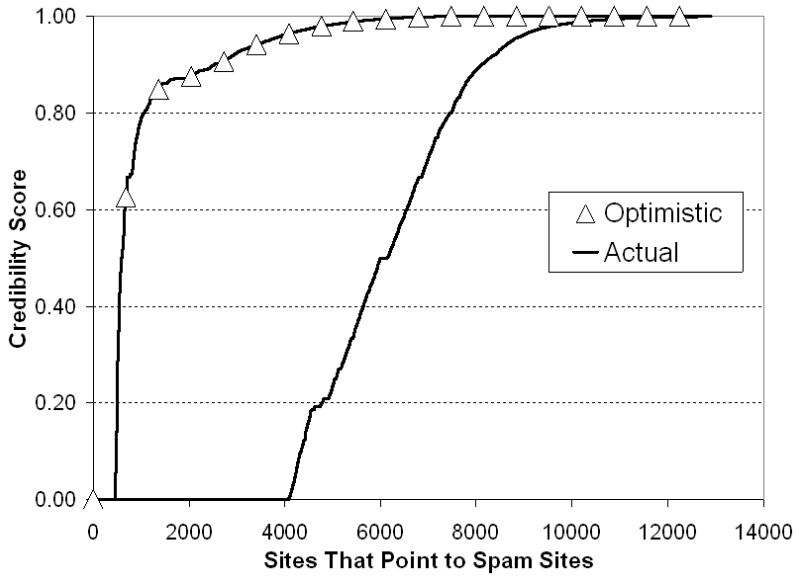

Figure 34: Optimistic Credibility Score Distribution (vs. Actual) [k=3]

credibility scores based on the entire Spam Corpus. The Optimistic approach assigns very high credibility for nearly all sites that point to the Spam Corpus, whereas the Pessimistic approach assigns 0 credibility to most sites. Only the Hop-Based approach balances these over and under estimation errors. As we will see in our spam resilience experiments in the following sections, this balance will lead to better spam-resilience than the Optimistic approach in all cases, and to better spam-resilience than the Pessimistic approach for $k > 2$.

The linear and exponential Hop-Based approaches are also impacted by the choice of the initial discount factor ψ. In Figure 37, we report the average credibility error for the linear case (for $L = 4$) for three setting of ψ (0.25, 0.50, and 0.75). It is encouraging to see that the error drops significantly for $k = 2$ and is fairly stable for increasing values of k (in contrast to the Constant and Pessimistic approaches reported in Figure 33).

We have also studied the impact of the partial blacklist size on credibility quality. We find that for the Optimistic, Linear, and Exponential approaches the error rate is fairly stable, whereas the Pessimistic and Constant approaches degrade severely as the blacklist size increases. For these two cases, a larger blacklist leads to more sites within a few hops of the blacklist, resulting in more 0 scores, even when the random walk probability of such a bad path is low.

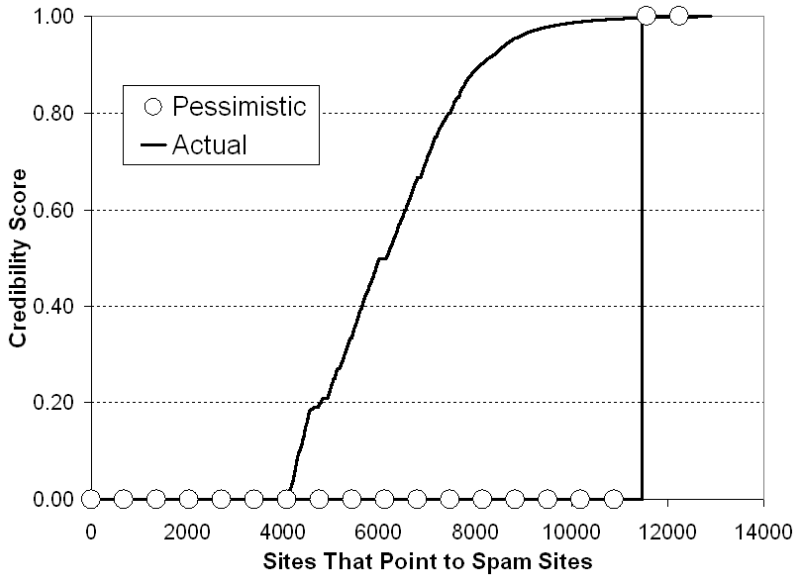

Figure 35: Pessimistic Credibility Score Distribution (vs. Actual) [k=3]

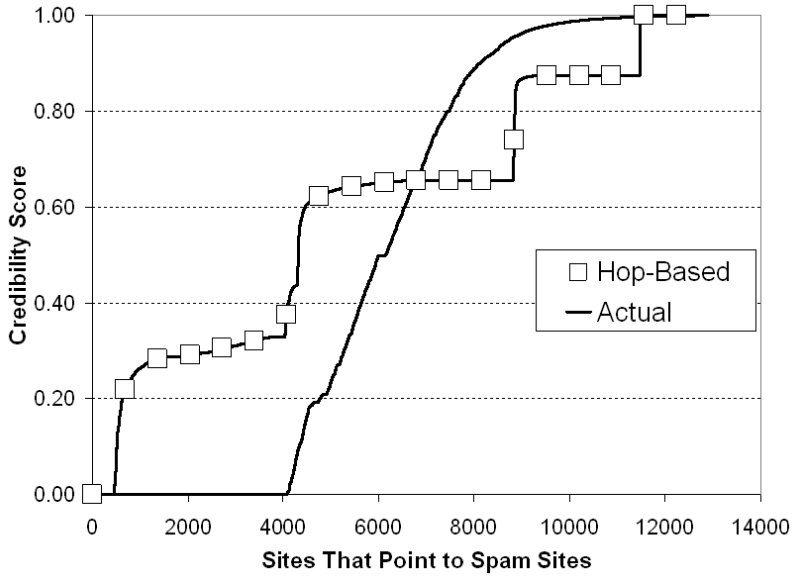

Figure 36: Hop-Based (exp $\psi = 0.5$) Credibility Score Distribution (vs. Actual) [k=3]

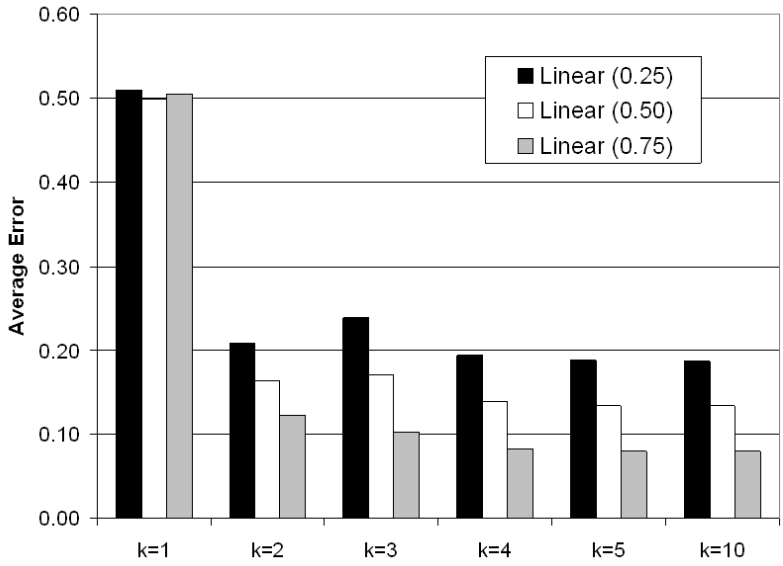

Figure 37: Average Credibility Error - Varying ψ

4.7.2.3 Time-Sensitive Credibility

In Section 4.4 we introduced several strategies for incorporating link history into credibility evaluation. In this set of experiments, we illustrate this time-sensitive credibility over two common spam scenarios. In Scenario 1, we consider a site that historically has had many links to spam sites, but then chooses to "whitewash" its credibility by removing all links to spam sites. In Scenario 2, we consider a legitimate site that has been hijacked by a spammer.

For Scenario 1, we randomly selected a non-credible site with a Hop-Based (exponential, $\psi = 0.5$, $k = 3$) credibility score less than 0.01. For this experiment, we assume that the site has historically always had spam links, and only in the most recent Web snapshot does it replace all of its spam links with links to reputable (non-spam highly credible) sites. In Figure 38, we report the time-sensitive credibility for the whitewashed site by considering the current "clean" snapshot and an increasing number of history snapshots as we vary the importance weight parameter λ. By considering the history component, we see that for all values of λ the credibility of the site achieves a maximum of 0.8. Note that $\lambda = 0.25$ favors the most recent credibility score for a page. Choosing $\lambda = 0.75$ places less emphasis on the most recent snapshot, and hence the credibility falls as low as 0.27 when 8 history

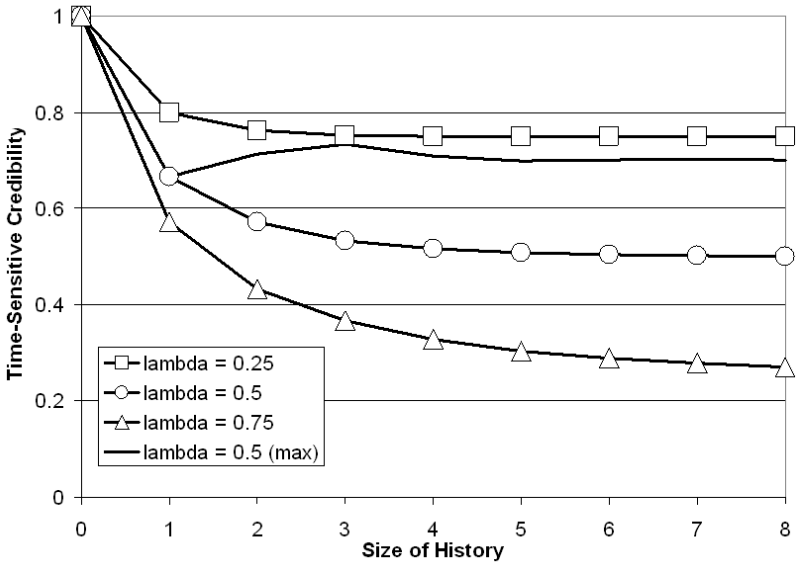

Figure 38: Time-Sensitive Scenario 1 (whitewash)

snapshots are considered. To illustrate the missing page policy, we randomly dropped four credibility measurements for the site. Using the *max* policy favors the most recent good snapshot, and so we see how the credibility score rises considerably.

For Scenario 2, we randomly selected a highly credible site with a Hop-Based (exponential) credibility score above 0.99. We then modified the Web graph to remove all of the site's existing links and inserted links to sites in the Spam Corpus, resulting in a new credibility score for the page of 0. In Figure 39, we investigate how quickly the site can recover its high credibility once it purges the spam links and replaces them with its original links. As the history component increases, more "clean" snapshots are available, and we can see how the site's credibility can quickly recover from a single hijacking incident.

4.7.3 Spam Resilience Evaluation

In the following sections we evaluate the quality of each credibility assignment approach through Web ranking experiments, and compare these results with PageRank and TrustRank. We measure the effectiveness of a Web ranking approach by its spam resilience. To quantify spam resilience, we introduce two metrics that each evaluates the quality of a candidate ranking algorithm's ranked list of sites versus a baseline ranking with respect to a set

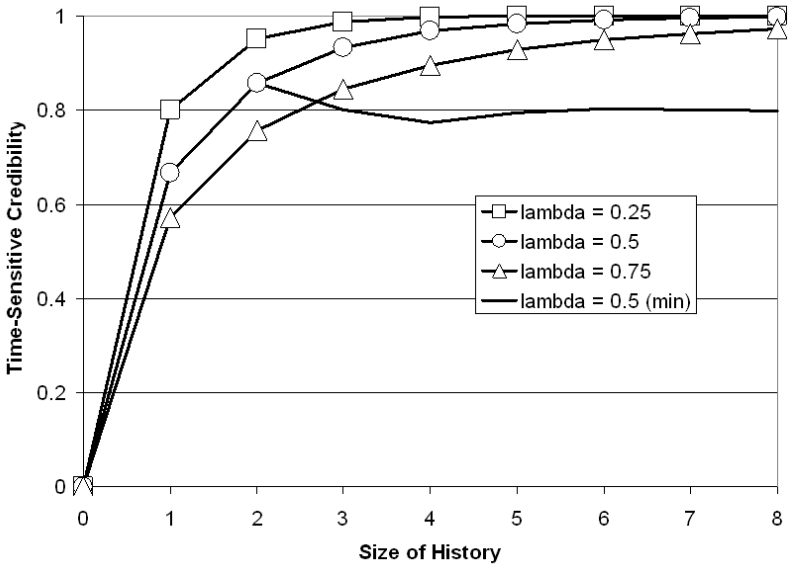

Figure 39: Time-Sensitive Scenario 2 (hijack)

of spam sites $\mathcal{X}$. We refer to $\mathcal{X}$ as a *portfolio* of spam sites. In this chapter, we use the Spam Corpus as the portfolio $\mathcal{X}$. We consider the baseline ranking for a portfolio of $|\mathcal{X}|$ sites: $B = (B_1, ..., B_{|\mathcal{X}|})$, and a ranking induced by the candidate ranking algorithm $E = (E_1, ..., E_{|\mathcal{X}|})$.

Rank-Based Spam Resilience

The first spam resilience metric SR_{Rank} measures the relative change of the *ranks* of the portfolio sites:

$$SR_{Rank}(m) = \frac{\sum_{i=1}^{m} R(E_i)}{\sum_{i=1}^{m} R(B_i)} - 1$$

where $R(E_i)$ returns the rank of site E_i according to the candidate ranking algorithm and $R(B_i)$ returns the rank of site B_i according to the baseline ranking algorithm. By evaluating $SR_{Rank}(m)$ for different values of m, we may assess the spam resilience of a ranking algorithm at different levels (e.g., for the top-100 pages, the top-1000, and so on). A candidate ranking algorithm that induces a ranking that exactly matches the baseline ranking will result in $SR(m)$ values of 0 for all choices of k. A ranking algorithm that induces a more spam-resilient ranking will result in positive $SR_{Rank}(m)$ values, meaning that the rank of

the portfolio will have been reduced. Negative values indicate that the candidate algorithm is less spam-resilient than the baseline.

Value-Based Spam Resilience

The second spam resilience metric is based on the change in *value* of the spam portfolio. Let us assume that each rank position has an associated value (say, in dollars), and that these values are monotonically decreasing as the rank position increases. That is, for a value function $V(\cdot)$, we have $R(i) < R(j) \Rightarrow V(R(i)) > V(R(j))$. Hence, we can measure the spam resilience by considering the relative change in the value of the spam portfolio under the candidate ranking algorithm versus the baseline ranking algorithm:

$$SR_{Value}(m) = 1 - \frac{\sum_{i=1}^{m} V(R(E_i))}{\sum_{i=1}^{m} V(R(B_i))}$$

where $V(R(E_i))$ and $V(R(B_i))$ are the values of applying a value function to the rank $R(E_i)$ and $R(B_i)$ respectively. A positive SR_{Value} value means that the candidate algorithm is more spam-resilient than the baseline algorithm since the overall value of the spam portfolio has been reduced. We consider a power-law rank value function since there is nice intuitive support for it, that is, the top-rank positions are quite valuable, while most positions have relatively low value. Confirming this intuition are several previous studies that have indicated that users tend to focus on the top-ranked results (e.g., [95, 107]), meaning that a very lowly-ranked page may be of little value. Concretely, the value of rank x is $V(x) = 1,000,000x^{-0.5}$, meaning that the top-ranked site has value of $\$1m$, the 100th-ranked site is worth $\$100k$, the 10,000th-ranked site is worth $\$10k$, and so on.[2] The key here is not to estimate the actual value precisely, but to provide a relative value of different rank positions.[3]

[2] Although PageRank, TrustRank, and CredibleRank scores are also distributed according to a power-law, it is inappropriate to rely on these scores to estimate value since these power-law distributions vary across ranking algorithms and parameter settings. We believe a value function is primarily dependent on rank position and fairly independent of the algorithm choice for determining rank positions.

[3] Recall that most ranking systems combine query-independent authority measures (like PageRank) with query-dependent features (like the presence and placement of particular keywords within a page) to generate their final rankings. In this chapter, we focus our analysis on query-independent ranking, so we consider a single global ranking (and value) over all sites.

4.7.3.1 PageRank versus CredibleRank

Given the above two metrics, we now compare the effectiveness of CredibleRank in comparison to PageRank with respect to spam resilience. Here PageRank is used as the baseline ranking. For fairness of comparison, we do not incorporate any whitelist information into the CredibleRank calculation, so the static score vector in Equation 9 is set to the uniform vector, as it is in PageRank (Equation 7).

For CredibleRank, we consider the Naive approach and three tunable k-Scoped Credibility assignment approaches – Optimistic, Pessimistic, and Hop-Based (exponential $\psi = 0.5$) – using the medium blacklist for scope of $k = 2$. In Figures 40 and 41, we report the $SR_{Rank}(m)$ and $SR_{Value}(m)$ spam resilience scores for $m = 1$ to $m = 9,034$ (the size of the Spam Corpus) for the four candidate CredibleRank rankings (i.e., Opt, Pess, Hop, and Naive) versus the baseline PageRank ranking. We are encouraged to see that for both rank-based and value-based spam resilience that all CredibleRank approaches result in more spam-resilient rankings versus PageRank, with the Pessimistic performing the best, closely followed by the Hop-Based approach. The spam resilience rapidly increases and then peaks over the top-2000 spam sites, indicating that CredibleRank performs well over these top-ranked spam sites. As k increases to consider more sites in the spam resilience measurement, more lower-ranked sites are considered which have less downward space to move, meaning that the overall spam resilience decreases relative to the top-ranked sites.

Changing the value function $V(x) = ax^b$ to consider different values of a has no impact on the spam resilience results since only relative value is measured. Secondly, changing the exponent b to $b = -0.1$ results in a less profound drop in value from one rank to the next, and we find that the $SR_{Value}(m)$ scores are in the 10% range. When $b = -0.9$ the value drop from one rank to the next is more severe, meaning that the top-ranked sites are valued very heavily; for this scenario, we again find that CredibleRank is more spam-resilient, with $SR_{Value}(m)$ scores in the 40-50% range.

To further demonstrate how CredibleRank demotes spam sites relative to PageRank, we sorted the sites by rank order for the Hop-Based CredibleRank and PageRank ranking vectors, and divided the sites into 20 buckets of an equal number of sites. Along the x-axis

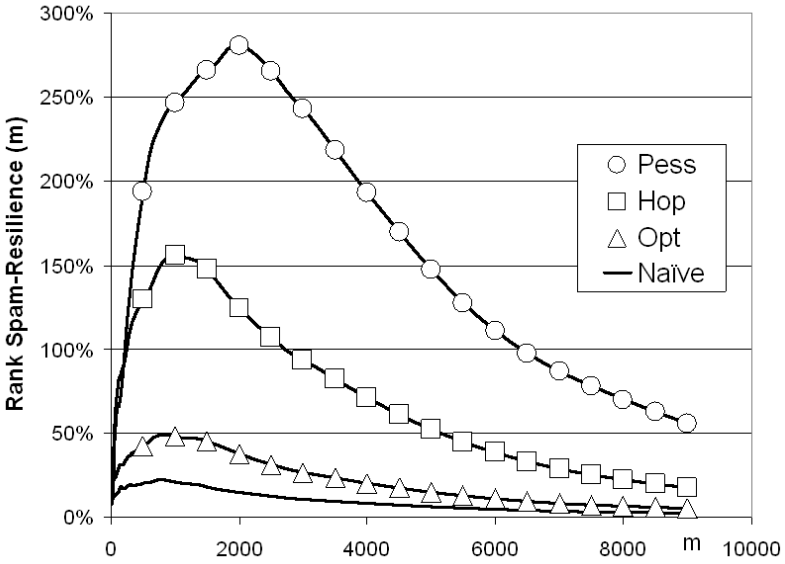

Figure 40: CredibleRank vs. PageRank: Rank Spam Resilience

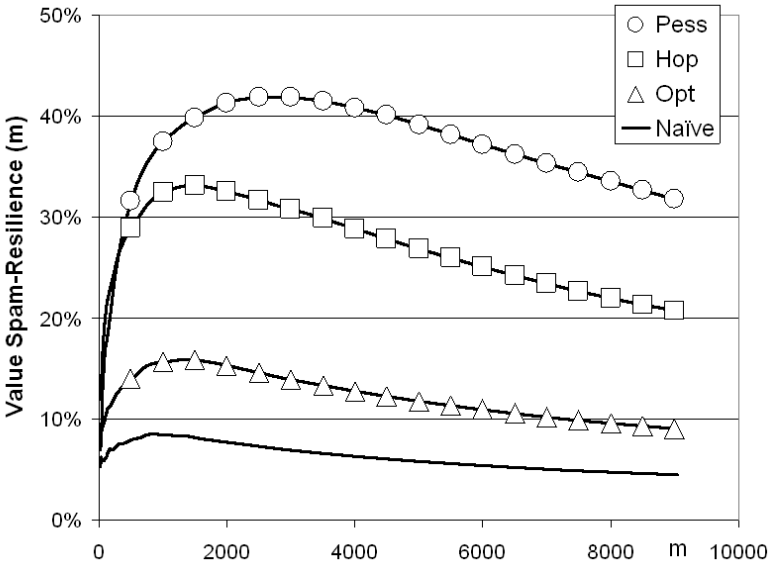

Figure 41: CredibleRank vs. PageRank: Value Spam Resilience

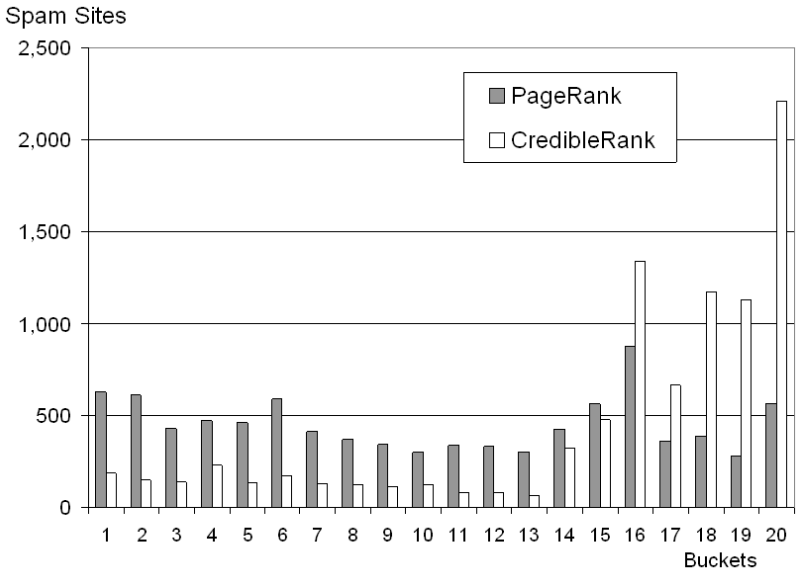

Figure 42: CredibleRank vs. PageRank: Spam Distribution

of Figure 42 we consider these 20 buckets, from the bucket of top-ranked sites (bucket 1) to the bucket of the bottom-ranked sites (bucket 20). Along the y-axis, we plot the number of Spam Corpus sites (of the 9,034 total spam sites) in each bucket. What is immediately obvious is that CredibleRank penalizes spam sites considerably more than PageRank by demoting spam sites to lower-ranked buckets, even when only 10% of the spam sites have been explicitly assigned to the blacklist.

Our results so far have measured the effectiveness of CredibleRank with respect to its spam resilience. We also would like to show that CredibleRank does not negatively impact known good sites. Hence, we compared the ranking of each whitelist site under PageRank versus its ranking under CredibleRank. We find that the average rank movement is only 26 spots, meaning that we can feel fairly confident that CredibleRank is not unduly punishing good sites.

4.7.3.2 TrustRank versus CredibleRank

Recall that TrustRank incorporates whitelist information into the ranking calculation to favor whitelist sites and the sites that they point to over other sites. In this experiment, we compare CredibleRank to TrustRank, where TrustRank is used as the baseline ranking

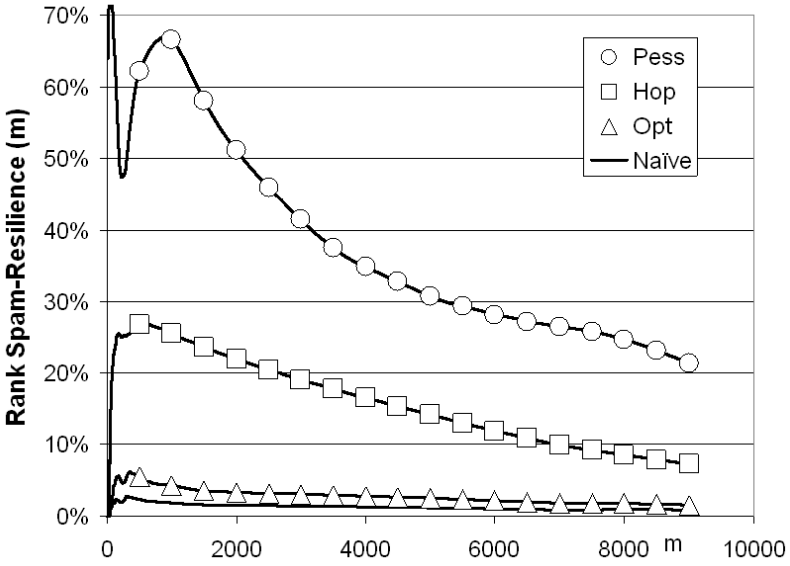

Figure 43: CredibleRank vs. TrustRank: Rank Spam Resilience

and for fairness, the CredibleRank approach relies on the same whitelist-based static score vector used in TrustRank (Equation 8).

For CredibleRank, we again consider the four link credibility assignment approaches – Naive, Optimistic, Pessimistic, and Hop-Based – using the medium blacklist. In Figures 43 and 44, we report the $SR_{Rank}(m)$ and $SR_{Value}(m)$ spam resilience scores for $m = 1$ to $m = 9,034$ for the three candidate CredibleRank rankings versus the baseline TrustRank ranking. As in the PageRank comparison, we see that all CredibleRank approaches result in more spam-resilient rankings comparing to TrustRank, with the Pessimistic and Hop-Based performing the best.

For the Hop-Based CredibleRank and the TrustRank ranking vectors, we report the bucket-based spam site distribution in Figure 45. We find that CredibleRank penalizes spam sites considerably more than TrustRank, pushing most sites into the bottom-ranked buckets.

We wish to note that the choice of whitelist is extremely important for TrustRank. Since links from whitelist sites are favored over links from other sites, a spammer has a great incentive to induce links from a whitelist site. In our experiments, we find choosing a whitelist with sites that either link directly to spam sites or are within several hops of spam sites

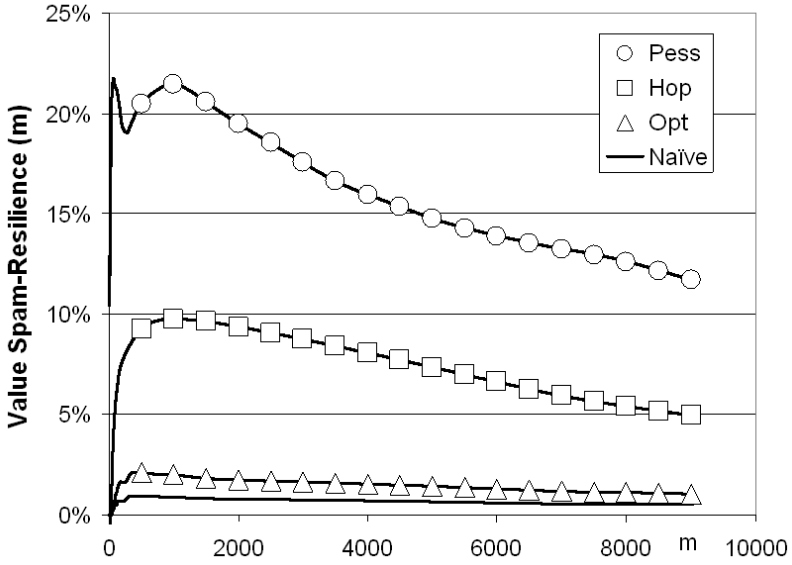

Figure 44: CredibleRank vs. TrustRank: Value Spam Resilience

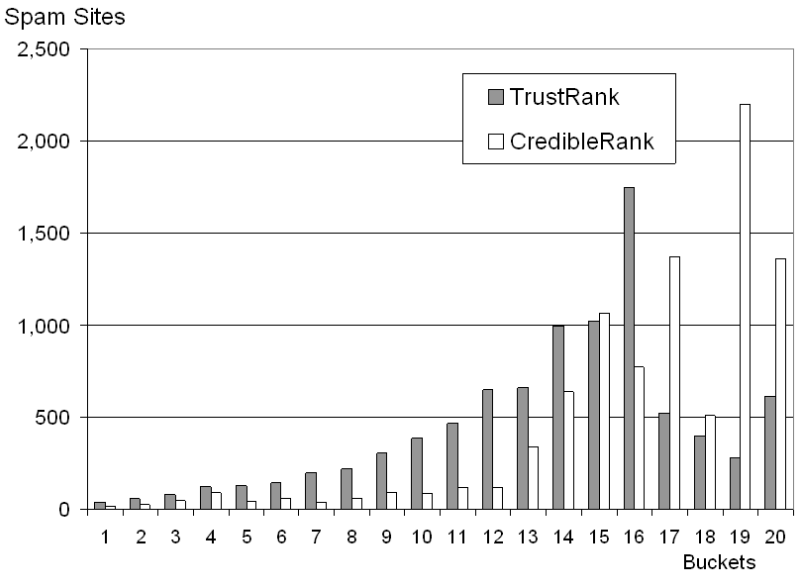

Figure 45: CredibleRank vs. TrustRank: Spam Distribution

results in very poor spam resilience for TrustRank. We find for one poor quality whitelist that CredibleRank has a rank-based spam resilience achieving a maximum improvement of 107% over TrustRank, with a 32% improvement over the entire spam corpus. We have also evaluated CredibleRank and TrustRank using only blacklist information and no whitelist information (by skewing the static score vector to non-blacklist sites). Since CredibleRank distinguishes page (or site) quality from link credibility, we find that it achieves rank-based spam resilience of up to 134% over TrustRank, with a 16% improvement over the entire spam corpus.

4.7.3.3 Impact of Scope (K)

Our results so far have measured the effectiveness of CredibleRank with respect to the tunable k-Scoped Credibility function for $k = 2$. But what are the implications of changing the scope parameter on the spam-resilience of CredibleRank? In Figure 46 we report the value-based spam resilience for $k = 1$ to $k = 5$ for the Naive approach and the three tunable k-Scoped Credibility assignment approaches – Optimistic, Pessimistic, and Hop-Based (exponential $\psi = 0.5$). The Naive approach does not consider scope and so its spam-resilience is unaffected by changes in k. The Hop-Based and Optimistic approaches are fairly stable with increasing k. For $k = 1$ and $k = 2$, the Pessimistic approach performs well, since sites that either directly link or are within 2 hops of blacklist sites have no ranking influence over the sites that they point to. The Pessimistic approach severely degrades in spam-resilience for increasing values of k until it performs even worse than PageRank for $k = 5$. When $k = 5$, nearly all sites have at least one path to a blacklist site, resulting in a Pessimistic credibility score of 0. In the CredibleRank interpretation, this means that nearly all links in the Web graph are disregarded and so the resulting rankings are essentially random.

4.7.3.4 Impact of Blacklist Size

We have also explored the impact of the blacklist size on the spam resilience of CredibleRank. For the three blacklists – small (1% of the Spam Corpus), medium (10%), and large (20%) – we report in Figure 47 the ranking distribution of the Spam Corpus for the Hop-Based

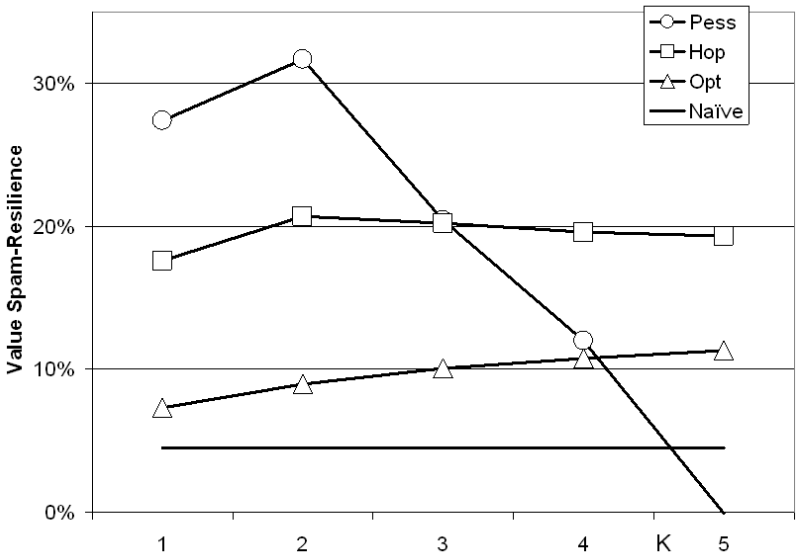

Figure 46: Impact of Scope [K] (CR vs. PR)

CredibleRank versus PageRank. For presentation clarity, we divide all sites into 10 buckets in this experiment. CredibleRank based on the small blacklist (containing just 90 sites) results in a remarkable improvement over PageRank. The advantage increases as more spam sites are added to the blacklist.

We also evaluated CredibleRank's spam-resilience versus TrustRank for varying choices of scope parameter (k) and blacklist size, and we find results of a similar spirit to the ones reported for PageRank in Sections 4.7.3.3 and 4.7.3.4.

4.7.4 Blacklist Expansion

In our final experiment, we report results for our credibility-based blacklist expansion algorithm described in Section 4.6. In Table 8 we report the precision of the blacklist expansion using a static score vector based on the exponential Hop-Based credibility scores used in our previous experiments. These results are based on the Medium blacklist.

Recall that the blacklist expansion algorithm ranks all sites with respect to spam-likelihood. Here we measure the precision of identifying as spam all sites ranked higher than some threshold (e.g., top-10, top-100, etc.). For example, in the top-1,000 sites as ranked by spam-likelihood, we find that 94.4% are actually spam. Of these 944 spam sites,

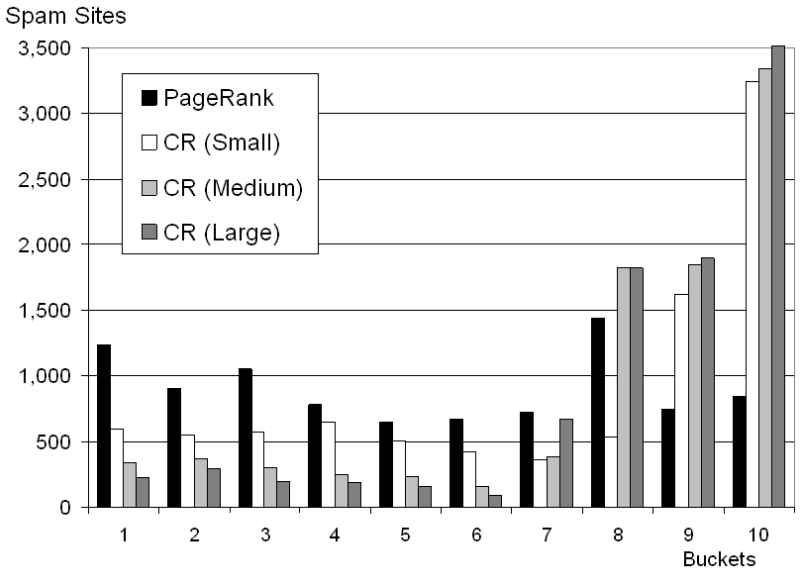

Figure 47: Impact of Blacklist Size (CR vs. PR)

Table 8: Blacklist Identification Precision

Rank Threshold	Precision	On Blacklist	Newly Identified
10	90%	1	8
100	89%	1	88
1000	94%	4	940
2000	91%	4	1814
3000	81%	11	2428
4000	70%	15	2785
5000	62%	15	3075
10000	44%	26	4414

only 4 were on the initial blacklist, meaning that we have identified 940 new spam sites previously unknown to us. Increasing the threshold results in a degradation of precision, and by the top-10,000, only 44% of sites are actually spam. In our continuing work, we are exploring whether a very tight threshold (say, the top-100) coupled with repeated applications of the blacklist expansion will perform even better.

4.8 Related Work

Our notion of link credibility has some analogues in trust network research, in which computational models are developed for measuring trust. One of the seminal papers in trust research [20] argued for distinguishing between direct trust and recommendation trust. In the context of transactional peer-to-peer networks, the PeerTrust system models the believability (or credibility) of peer feedback to guide the trust calculation of nodes in the network [183]. Link credibility is also somewhat related to the notion of distrust, which has recently received increasing attention (e.g., [75], [179]). For example, in [75], the authors argue for a trust propagation technique in which the recommendations of distrusted nodes are discounted completely. Note that our link credibility model allows for a continuum of credibility scores.

4.9 Summary

In this chapter, we have explored the concept of link credibility, presented several techniques for semi-automatically assessing link credibility for all Web pages, and presented an efficient and yet spam-resilient credibility-based Web ranking algorithm. We also introduced a set of metrics to measure the spam resilience properties of credibility-based link analysis, and have shown that our credibility-based ranking algorithm outperforms both PageRank and TrustRank.

CHAPTER V

TRUST ESTABLISHMENT IN ONLINE COMMUNITIES

In the previous two chapters, we focused on vulnerabilities in link-based search services. We now move to the second of our three major topics, wherein we study tamper-resilience in trust-based algorithms for online communities. Online communities like Bebo, Facebook, MySpace, and Xanga have grown tremendously in the past few years, enabling millions of users to discover and explore community-based knowledge spaces and engage in new modes of social interaction. These online communities are positioned as a potential future computing platform; in fact, Facebook has recently released an API for supporting a "social operating system" [165]. This opportunity and growth has not come without a price, however, as we explored in Chapter 2. Online communities have been targeted for malware dissemination [27, 102], identify theft [42, 94], and deception in digital identity [155], to name just a few threats.

With these problems in mind, we focus on building an online community platform that allows wide access to many different types of users and that still remains useful, even in the presence of users intent on manipulating the system. We introduce SocialTrust, a social network trust aggregation framework for supporting tamper-resilient trust establishment in online social networks. A trust-based approach is one of the most promising avenues for maintaining the relative openness of these communities (and the corresponding benefits) and still providing some measure of resilience to attacks. Using trust ratings, a user can decide with whom to engage in new social interactions, to form new groups, to engage in transactions, and so forth. SocialTrust establishes trust for a participant in the social network through four key features – (i) the trust establishment scope in which we build trust; (ii) how well the user participates in the network; (iii) the quality of a user's relationships in the network; and (iv) the history of the user's behavior.

The Need for Trust? An important point that deserves attention is whether trust is

necessary at all. The *no trust* case is widespread in popular online social networks like MySpace and Facebook, where users maintain only their local relationships with no access to system-wide or aggregated trust information. The advantage of such an approach is the avoidance of potentially expensive trust computations and the sharing of user relationship links with either other users or the system as a whole. The lack of such aggregated trust information has certainly not stymied the growth of these networks, but as these communities attain more mass and public attention, we observe an increasing number of attacks on the legitimacy of these communities.

In summary, this chapter makes the following contributions:

- We study online social networks, consider a number of vulnerabilities inherent in online social networks, and introduce the SocialTrust framework for supporting tamper-resilient trust establishment.
- We study four key factors for trust establishment in online social networks – trust establishment scope, trust group feedback, and relationship link quality – and describe a principled approach for assessing each component.
- In addition to the SocialTrust framework, which provides a network-wide perspective on the trust of all users, we describe a personalized extension called mySocialTrust, which provides a user-centric trust perspective that can be optimized for individual users within the network.
- We experimentally evaluate the SocialTrust framework using real online social networking data consisting of millions of MySpace profiles and relationships. While other trust aggregation approaches have been developed and implemented by others, we note that it is rare to find such a large-scale experimental evaluation that carefully considers the important factors impacting the trust framework.

In the following section, we begin by modeling online social networks (Section 5.1). We identify several vulnerabilities in Section 5.2 and discuss strategies for countering these vulnerabilities in Section 5.2. The overall SocialTrust framework is presented in Section 5.3.

We then provide an in-depth look at computing feedback ratings in Section 5.5, link quality in Section 5.6, and trust in Sections 5.7 and 5.8. We present the personalized MySocialTrust approach in Section 5.9. We describe the experimental setup in Section 5.10 evaluate the trust model in Section 5.11, before wrapping up with related work in Section 5.12, and a summary in Section 5.13.

5.1 Reference Model

In this section, we describe the reference model for an online social network. The model represents an online social network $\mathcal{SN}$ as a triple consisting of profiles $\mathcal{P}$, relationships $\mathcal{R}$, and contexts $\mathcal{C}$: $\mathcal{SN} =< \mathcal{P}, \mathcal{R}, \mathcal{C} >$. A profile p is the online representation of a particular person, place, or thing. We denote the set of all profiles in the social network $\mathcal{SN}$ as $\mathcal{P}$. We shall assume there are n profiles in the network, numbered from 1 to n: $\mathcal{P} = \{p_1, ..., p_n\}$. We denote a relationship between profiles i and j with two entries in the relationship set $\mathcal{R}$ to characterize each participant's contextual view of the relationship: $rel(i, j, c_1)$ and $rel(j, i, c_2)$, where c_1 and c_2 are two contexts drawn from the context set $\mathcal{C}$. We denote user i's set of friends as $rel(i)$ and the total number of relationships i participates in as $|rel(i)|$.

The most basic element in a social network is a *profile*. We most typically think of profiles in terms of specific people (e.g., me, my friends, my parents, etc.), but profiles may represent companies, products, musical groups, abstract concepts (like love, harmony, and peace), and so on. For simplicity in presentation, we will refer to profiles and users interchangeably in the rest of this chapter. Typically, a profile is a user-controlled Web page that includes some descriptive information about the person it represents. In Figure 48, we illustrate a sample profile from Facebook. Most profiles include some personal information like the person's picture, age, gender, location, and interests. Additionally, a profile may be augmented with a collection of digital artifacts like Web documents, media files, etc.

Profiles are connected to other profiles through explicitly declared *relationships*. A relationship in a social network is a bidirectional link between two users. A relationship is only established after both parties acknowledge the relationship. To initiate a relationship, a user sends a request to another user. If the other user accepts this request, the relationship

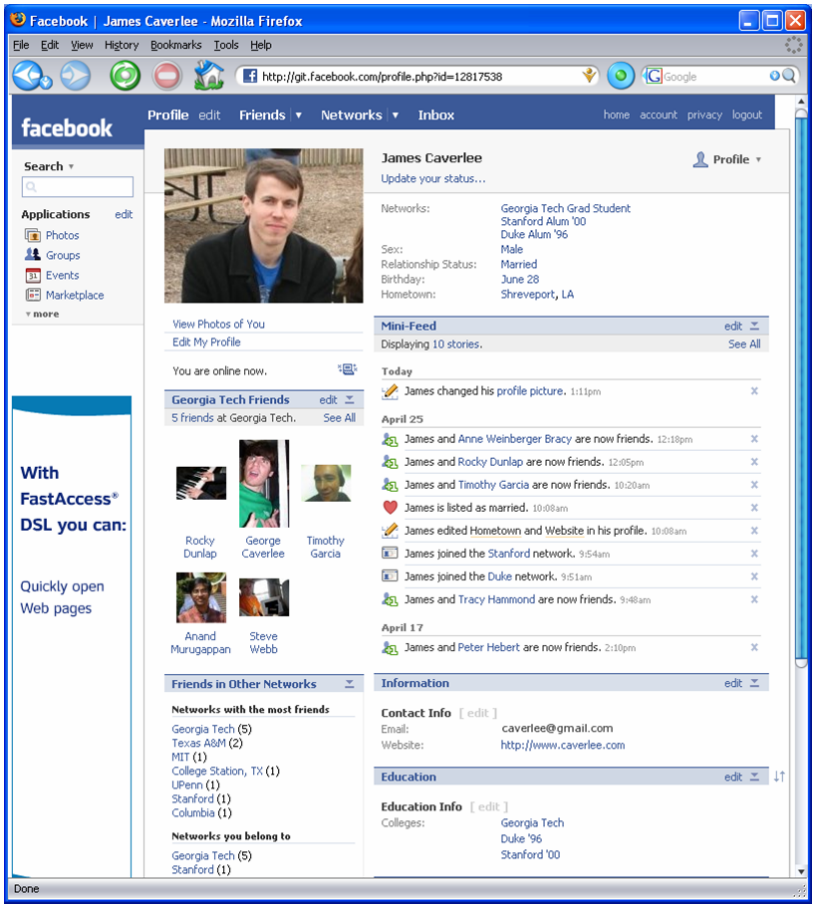

Figure 48: Sample Profile from Facebook

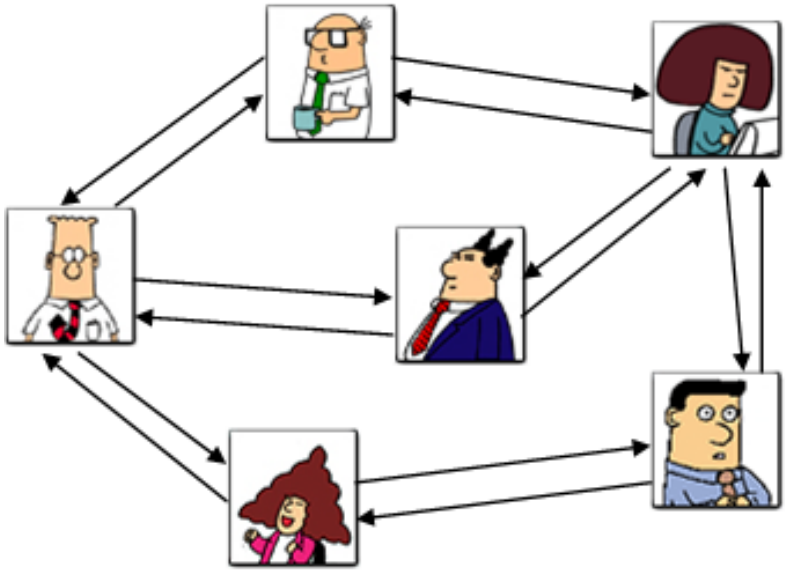

Figure 49: Simple Social Network: Profiles and Relationship Links

is established, and a relationship link is added to both users' profiles.

A relationship link may be augmented with *contextual* information indicating the nature of the relationship – e.g., the two people are friends from school, co-workers, neighbors, etc. The context is an annotation of a relationship. Several recent ethnographic studies of online social networks have identified a number of reasons for friend formation in online social networks, including: the two nodes are actually friends, as a means of displaying popularity, for friend "collecting", for reasons of courtesy, and others [28, 147]. We emphasize that we make no requirements on how relationships in the community arise, and all of the algorithms presented in this chapter are agnostic to this relationship formation.

We can view the social network $\mathcal{SN} =< \mathcal{P}, \mathcal{R}, \mathcal{C} >$ as a graph where the profiles $\mathcal{P}$ are nodes and the relationships $\mathcal{R}$ are labeled directed edges. A node in the graph represents one profile. Each node in the graph is denoted by the associated profile's number. A labeled directed edge in the graph represents a relationship link from one profile to another. A relationship link from profile i to j is represented by the edge from node i to node j in the graph and is denoted with a context c as $i \xrightarrow{c} j$. In Figure 49, we illustrate a simple network of six profiles and their declared relationships.

5.2 Vulnerabilities in Online Social Networks

In Section 2 we identified a number of threats to the privacy and security of online communities. While there are important problems associated with securing the social network infrastructure (e.g., ensuring that profiles are correctly formatted and contain no browser exploits, containing denial-of-service attacks), we explore vulnerabilities to the quality of information available through online social networks even when the underlying social network infrastructure has been secured. We make note of some key differences between online social networks and the Web-at-large.

In particular, we identify three important vulnerabilities:

- **Malicious Infiltration:** Unlike the Web-at-large, most online social networks do provide some limits as to who can and cannot participate. Users typically must register with an online social network using a valid email address or by filling out a user registration form. As a result, many social networks give the illusion of security [14], but malicious participants can gain access as has been observed in MySpace [155] and the more closely guarded Facebook [32].

- **Nearby Threats:** Unlike the Web domain, in which a site can link to any other site on the Web, online social networks enforce bilateral agreement between both parties in a relationship. As a result, participants in a social network have tight control over who their friends are. There is a well-noted illusion of privacy in online social networks [29, 136] in which participants have a lack of understanding of the potential threat of participants two or more hops away. The small world phenomenon – a feature of many social networks [126, 175, 176] – means that there is a short distance in the network between any two participants. So even if a user has tight control over his direct friends, malicious participants can be just a few hops away from any participant. For example, in Chapter 2 (Figure 7), we identified a deceptive rogue profile on MySpace with nearly 200 declared friendships (see Figure 50) with (seemingly) legitimate community members, indicating that some community members are either easily deceived or have low standards for friendship.

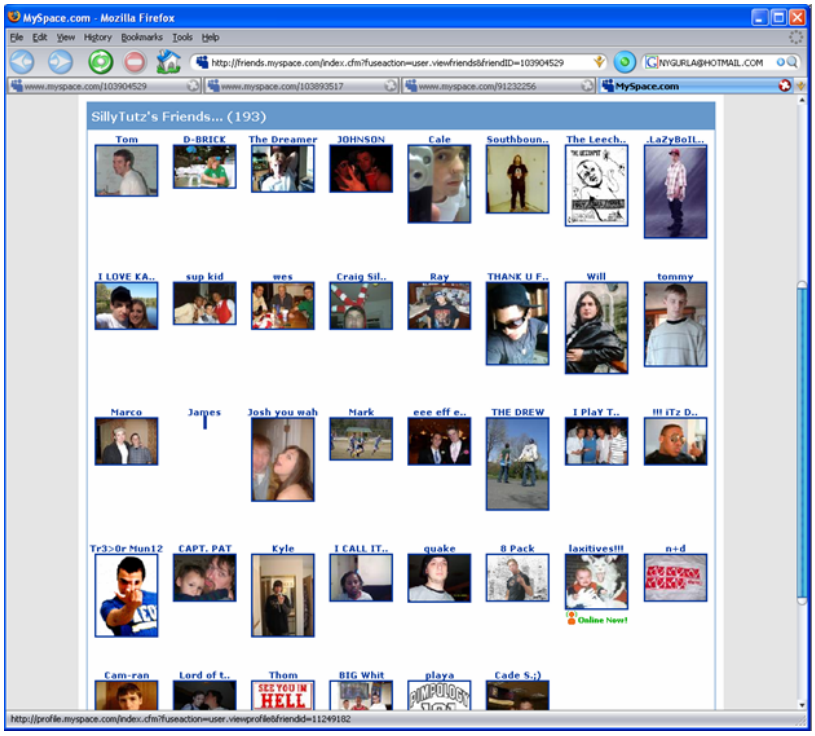

Figure 50: Example Deceptive Profile with Many Legitimate Friends on MySpace

- **Limited Network View:** Even if a user in the social network maintains tight control over her friends *and* closely monitors the quality of her neighbors' friends, she will still have access to only a limited view of the entire social network. Since any one user may have direct experience or relationships with only a small fraction of all social network members, she will have no assurances over the vast majority of all participants in the network.

To counter these vulnerabilities, we can rely on a number of approaches including legal enforcement, background checks, and reliance on a trusted central authority. A strictly legal approach is one adopted by many of the popular social networking sites, in which participants who are deemed to be in violation of the terms of service may be identified and removed from the social network. Complementary to this approach, a trusted central authority may be empowered to monitor the community (as in MySpace and Facebook). Alternatively, users may be required to undergo a background check to provide some off-line assurances as to their quality. These approaches typically suffer from problems with

enforcement and scalability. In this chapter, we use a *reputation-based trust approach* for maintaining the relative openness of these communities (and the corresponding benefits) and still providing some measure of resilience to the described vulnerabilities.

Many e-marketplaces and online communities use reputation systems to assess the quality of their members, including eBay, Amazon, and Digg, and reputation-based trust systems have received considerable attention in P2P systems (e.g., [1, 45, 48, 98]), as well as for facilitating trust in the Semantic Web [148]. These approaches aggregate community knowledge for evaluating the trustworthiness of participants. The benefits of reputation-based trust from a user's perspective include the ability to rate neighbors, a mechanism to reach out to the rest of the community, and some assurances on unknown users in the network. Most existing approaches, however, ignore the social constructs and social network topology inherent in online social networks, and typically provide less personalized criterion for providing feedback and computing reputations. A key challenge then is whether we can develop a trust framework for online social networks that is tamper-resilient even in the presence of malicious users. And what are the key factors impacting such a framework?

5.3 The SocialTrust Framework

In this section, we introduce the SocialTrust framework. The goal of SocialTrust is to provide a trust rating for each user in the online social network by leveraging the rich social connections of the social network. SocialTrust is explicitly designed to (i) leverage the relationships inherent in the social network; (ii) gracefully handle the addition of new users to the network as it evolves; and (iii) be robust against efforts to manipulate the trust ratings.

SocialTrust establishes trust for a participant in the social network through four key features – (i) the trust establishment scope in which we build trust; (ii) how well the user participates in the network; (iii) the quality of a user's relationships in the network; and (iv) the history of the user's behavior.

We denote the SocialTrust trust rating of user i by $Tr(i)$. For any two users in the community, we may evaluate the relative trustworthiness, e.g., that user i is more

trustworthy than user j (i.e., $Tr(i) > Tr(j)$). This aggregated trust information may be used by users for enhancing the quality of their experiences in the community. Since users will typically have direct relationships with only a small fraction of all users in the network, trust values may be used to evaluate the quality of the vast majority of other users for which the user has no direct experience.

For presentation clarity, we shall assume the presence of a centralized *trust manager* whose job is to compute trust ratings for users in the network and to communicate these trust ratings to users when needed. Alternatively, the duties of the trust manager may be securely distributed throughout the network (see, for example, [96]).

Initially all users are treated equally. SocialTrust supports trust maintenance through dynamic revision of trust ratings according to four critical components:

- **Trust Establishment Scope:** Trust establishment scope governs which other participants in the social network a user can make an assessment of (and conversely, which other participants can make an assessment of that user).

- **Trust Group Feedback:** The second key component of trust establishment is the feedback rating of participants in the network. User i's feedback rating $F(i)$ could be used directly for trust establishment, but it takes no advantage of the rich social connections of the online social network for evaluating user trustworthiness.

- **Relationship Link Quality:** Hence, the third component of trust establishment for a user in the social network is that user's relationship link quality, denoted $L(i)$. Monitoring link quality provides an incentive for users to ensure the quality of their relationships.

Based on these first three components, we can establish the overall quality component of trust $Tr_q(i)$ for the user.

- **History Component of Trust:** The final component of SocialTrust is the history component $Tr_h(i)$, which considers the evolution of a user's trust rating. This history

component is important for limiting the ability of malicious participants to whitewash their trust ratings by repeatedly leaving and re-entering the network.

The overall SocialTrust trust metric for user i is a simple linear combination of the quality component and the history component of trust:

$$Tr(i) = \alpha Tr_q(i) + (1 - \alpha) Tr_h(i) \tag{10}$$

where $0 \leq \alpha \leq 1$. Choosing $\alpha = 1$ favors the most recent quality assessment and disregards the history component altogether. Conversely, choosing $\alpha = 0$ favors the history component over the most recent quality assessment.

Given the basic SocialTrust framework, there are a number of open questions. How is each component assessed? How are these components combined? What are the important factors impacting trust assessment? In the following sections, we address each of these questions to provide a thorough understanding of SocialTrust and how it supports robust trust establishment.

5.4 Trust Establishment Scope

The trust establishment scope governs what other participants in the network each user can judge, and what other participants can judge each user. Trust group formation can be tuned to balance efficiency and scalability and the security of the overall system (by constraining users from manipulating the reputation of distant users).

At one extreme, there is a single *trust group* consisting of all members of the social network. At the other extreme, each user belongs to a lone trust group consisting of only themselves, meaning that the system supports no trust aggregation. For balancing these two extremes, we could rely on trust groups defined by self-described interests (e.g., sports), location (e.g., members who all live in Texas), or other contextual information.

We define trust groups based on the chains of relationships that are fundamental to the formation of social networks. Hence, we consider a *relationship-based* model for determining a user's trust group where the size of the trust group is determined by a network-specified

radius, ranging from a user's direct neighbors (radius 1), to a user's direct neighbors plus his neighbors' neighbors (radius 2), and so on.

Relationship-based trust groups can be formed either through the *browse-based search capability* or the *forwarding-based search capability* of online social networks.

5.4.1 Trust Group Formation: Browse-Based Search

Most current online social networks support browse-based search. By browse-based search capability, we mean that a user's profile may be viewed (or browsed) by other users and possibly the overall social network system (e.g., a central authority like in MySpace). Users may manually browse from profile to profile and provide ratings on users encountered subject to the degree of the relationship-based trust group. A user can elect to make portions of the profile either (i) public; (ii) semi-public; or (iii) private. A public profile can be browsed by all other members of the social network. A semi-public profile can be browsed by some other members of the social network, say direct friends. A private profile can be browsed by no other members. The browse-based search capability is inherently passive – a user need only agree to allow other users (or the system) to view portions of her profile.

5.4.2 Trust Group Formation: Forwarding-Based Search

Alternatively, in *forwarding-based search capability*, a user agrees to participate in the forwarding of queries through the social network along relationship links. This type of active search capability is similar in spirit to search in unstructured peer-to-peer networks, and we see it as a natural addition to current social networking sites for providing more personalized search functionality. Forwarding-based search proceeds as follows. An *originating user* sends a query to some subset of her friends, whose selection may depend on the context of their relationship. Each of the selected friends can choose to answer the query or to pass the query along to some subset of his neighbors, and so on, until a stopping condition is met. The originating user can rate each user who responds.

Within the entire trust group, a user may choose to send a query to the entire trust group or to some subset of the entire group. We consider three query forwarding models for exploring a user's trust group: Flooding, Random Selection, and Selective Forwarding.

- **Flooding**: For flooding, a user forwards a query to all of her neighbors, regardless of the content of the query or the expected quality of the neighbors in answering the query. Such a flooding mechanism can communicate the query to a large audience in the user's trust group, but at the expense of burdening many users.

- **Random Selection**: In the second case, a user forwards a query to some random selection of r friends. For user j with relationship set $rel(j)$, the user randomly selects $max(r, |rel(j)|)$ neighbors. For $r = 1$, this random selection is a simple depth-first traversal of the user's trust group. Such a query forwarding mechanism can be tuned to avoid the high message cost of flooding-based search, but at the expense of not finding high quality users for responding to the query.

- **Selective Forwarding**: In the third case, a user selectively forwards queries based on a matching criterion between the query and the user's neighbors. For now, we consider a fuzzy match function based only on the profile published by each user in the social network. Other features like previous search history, length of time in the network, and context-sensitive information could also be used for selecting neighbors. A user selects the best r scoring neighbors based on the match estimation.

We consider a *horizon-based* model for determining how wide a query will be forwarded within a user's trust group. Each query is annotated with an integer-valued *horizon* indicating how many hops through the network a query should pass, such that the horizon is not greater than the radius of the user's relationship-based trust group. When a user receives a query, it decrements the horizon value. When the horizon is 0, a user no longer forwards the query.

5.5 Assessing Trust Group Feedback

Given a trust group, we next describe several strategies for assessing the second component of SocialTrust – trust group feedback. We assume that each user i in the network is associated with a feedback value $F(i)$ that indicates how well the user's trust group views the user. The feedback ratings are taken from the interval $[0, 1]$. We make two observations:

(i) user behavior is dynamic, so the feedback ratings should be dynamically updated; and (ii) malicious users may attempt to subvert the feedback ratings.

For assessing feedback ratings, each user maintains state about the other users it has made a rating for (either through browsing-based search or forwarding-based search). As one example, in forwarding-based search an originating user i can rate user j based on well user j satisfies the query. Based on the ratings of all users who have interacted with user j, we can assess a feedback rating $F(j)$. Guaranteeing that feedback ratings are robust to manipulation is an important feature, and there have been several recent studies on how to ensure such robustness, e.g., [152], [163], and [183].

We briefly describe five important factors (as identified in [183]) that any feedback mechanism should consider (where we adapt these factors to online social networks):

- **Satisfaction:** When user i encounters user j either through the browsing process or through a query response, how well is user i satisfied with user j? This satisfaction level is the basis for assessing user j's overall feedback rating.

- **Number of Interactions:** For how many interactions does a user satisfy the originating user? For example, if user i satisfies 10 queries in the most recent period, but user j satisfies 999 out of 1,000, which user should be deemed of higher quality? A clique of malicious participants may engage in fake interactions with each other to mask their poor behavior to legitimate users in the network.

- **Feedback Credibility:** How credible are the users providing the feedback? Some users may habitually provide non-truthful feedback ratings while others are truthful. It is important to understand each user's credibility to prevent gaming of the feedback system through dishonest feedback.

- **Interaction Context Factor:** Some interactions between users may be more important than others, and the interaction context factor is intended to capture this relative value. For example, a user who provides high-quality job lead information for a high-ranking executive could be rewarded more in terms of feedback rating than a user who provides less valuable information.

- **Community Context Factor:** Finally, the online community may choose to reward members who provide feedback as an incentive for encouraging participation in the feedback process.

Based on these observations, we develop a feedback aggregation approach. We begin with a very basic rating system. A vote is a pair of the form $< user, vote >$, where *user* is a unique user identifier (the profile number) and *vote* is either "good" or "bad". Each user communicates to the trust manager a vote for each search in the most recent period. We consider three voting schemes – (i) open voting; (ii) restricted voting; and (iii) trust-aware restricted voting. We describe the first two and their drawbacks to motivate the final trust-aware restricted voting scheme. This final voting scheme incorporates the first three important factors described above; we anticipate revisiting it in our future work to customize it both in terms of search context factor and community context factor. Alternative feedback approaches are possible and easily pluggable into the SocialTrust framework.

5.5.1 Open Voting

We use the shorthand $v_i(j)^+$ to indicate a "good" vote by user i for user j; $v_i(j)^-$ indicates a "bad" vote. In the simplest case user j's feedback rating $F(j)$ is the fraction of "good" votes to the total votes cast for user j:

$$F(j) = \frac{\sum_i \mathcal{I}(v_i(j)^+)}{\sum_i \mathcal{I}(v_i(j)^+) + \mathcal{I}(v_i(j)^-)}$$

where the indicator function $\mathcal{I}(\cdot)$ resolves to 1 if the argument to the function is true, and 0 otherwise. This open voting policy is subject to ballot stuffing. A single malicious user can issue an unlimited number of "good" votes for raising the feedback rating of colluding users or can issue "bad" votes for demoting the feedback rating of competing users.

5.5.2 Restricted Voting

We can restrict how much each user can vote by assigning each user a limited number of *points* to be allocated over all of its votes. We let w_{ij} denote the number of points user i uses

to weight her vote for user j, where the total points allocated to each user is an arbitrary constant: $\sum_j w_{ij} = 1$. Hence, this restricted voting leads to a new feedback rating:

$$F(j) = \frac{\sum_i w_{ij}\mathcal{I}(v_i(j)^+)}{\sum_i w_{ij}\mathcal{I}(v_i(j)^+) + w_{ij}\mathcal{I}(v_i(j)^-)}$$

By restricting the total size of vote allocated to each user, this restricted voting scheme avoids the problem of vote stuffing by a single user. We have no assurances that a malicious user will choose to vote truthfully for other users it has actually interacted with, but we do know that the total amount of voter fraud is constrained. Unfortunately, such a voting scheme is subject to collusive vote stuffing, in which many malicious users collectively decide to boost or demote the feedback rating of a selected user.

5.5.3 Trust-Aware Restricted Voting

To handle the problem of collusive vote stuffing, we advocate a weighted voting scheme in which users are allocated voting points based on how trustworthy they are. We again let w_{ij} denote the number of points user i uses to weight her vote for user j, but now the total points allocated to each user depends on her trustworthiness: $\sum_j w_{ij} = Tr(i)$. This trust-aware restricted voting scheme results in a feedback rating for user j of:

$$F(j) = \frac{\sum_i Tr(i)w_{ij}\mathcal{I}(v_i(j)^+)}{\sum_i Tr(i)w_{ij}\mathcal{I}(v_i(j)^+) + Tr(i)w_{ij}\mathcal{I}(v_i(j)^-)}$$

Hence, feedback ratings and trust are connected from period to period. If a malicious user receives poor feedback from more trusted users in the system, then his feedback rating will be negatively affected, which in turn will impact his trustworthiness in the system. Intuitively, this cycle is appealing since it can dynamically adapt to trusted users who over time begin behaving badly as well.

5.6 Assessing Relationship Link Quality

In this section, we discuss the third critical factor of the SocialTrust framework – relationship link quality. Recall that user i participates in a total number of relationships $|rel(i)|$. How many of these relationships are with high quality users? Are any of these

relationships with users that have been flagged by the system? Our goal in this section is to formally assess the quality of a user's relationship links. Concretely, let $L(i)$ denote the *relationship link quality* of user i. A score of $L(i) = 0$ indicates that user i has poor quality relationship links. In contrast, a score of $L(i) = 1$ indicates that user i has high quality relationship links.

The small world nature of many social networks means that a large portion of the network may be reachable from any one user within a few hops. Hence, a user's relationship link quality should depend on the user's direct relationship links and perhaps the relationship links of its neighbors up to some small number (k) of hops away. We also observe that a user's link quality should be related to the feedback ratings of its neighbors. A user who only engages in relationships with well-behaving users should earn a higher link-quality score than a user who has relationships with poorly behaving members of the network. In the rest of this section, we formally define link quality and provide a discussion of the factors impacting its assessment.

5.6.1 Link Quality as a Scoped Random Walk

We model the link quality of user i in terms of a scoped random walk model, in which a random walker originates its walk at user i and randomly follows the relationship links of user i and the subsequent users at which it arrives up to some small number of steps.

In the extreme, when all users within k hops of the original user i have a perfect feedback rating (i.e., $F(j) = 1$ for all users within k hops of i), then user i has link quality $L_k(i) = 1$. In contrast, if user i either has a poor feedback rating (i.e., $F(i) = 0$) or all of the users within k hops of user i have poor feedback ratings, then user i's link quality is $L_k(i) = 0$. To summarize, the link quality of user i can be interpreted as the probability that a random walker originating its walk at node i ends at a high-quality user after walking up to k-hops away from i.

We can begin our examination of link quality by considering the base case when the scope (k) is 0.

Base Case (k=0): In the base case, the link quality of a user is merely its feedback rating

$F(i)$:

$$L_{[0]}(i) = F(i)$$

The random walker walks for 0-hops, meaning that it stays at the original user. The probability that the random walker ends at a high-quality user is thus $F(i)$.

One-Hop Case (k=1): In the one-hop case, the link quality of a user is the probability that the random walker ends at a high-quality user after walking forward to one of the friends from the original user's set of relationships (recall that $rel(i)$ denotes the relationship list for user i):

$$L_{[1]}(i) = F(i) \sum_{j \in rel(i)} F(j)/|rel(i)|$$

Note that the random walker proceeds initially according to the feedback rating $F(i)$ of the original user. Accordingly, the link quality of a user that have received poor feedback will be low. But a user with a high feedback rating who recommends poor quality users will be penalized with a low link quality.

Two-Hop Case (k=2): The link quality can be extended to consider random walks of length two, where:

$$L_{[2]}(i) = F(i) \sum_{j \in rel(i)} F(j)/|rel(i)| \left[\sum_{l \in rel(j)} F(l)/|rel(j)| \right]$$

We can extend link quality to consider random walks of arbitrary length k. In all cases, link quality is a local computation and can be updated in a straightforward fashion.

5.6.2 Correction Factor

The scoped random walk provides a natural measure of the relationship link quality of each user. Recall that the feedback ratings used for driving the link quality assessment may not be known with certainty and malicious users may attempt to subvert them. Hence, in this section, we discuss several correction factors for augmenting the basic scoped random walk model in the presence of such uncertainty. Such a correction factor is inspired by the

penalty factor associated with the tunable k-Scoped Credibility introduced in the previous chapter (recall Chapter 4).

We denote the updated link quality score for user i as $\hat{L}_{[k]}(i)$, and evaluate it in terms of the original link quality score and a correction factor ϕ:

$$\hat{L}_{[k]}(i) = \phi \cdot L_{[k]}(i)$$

We present an optimistic and a pessimistic correction factor as two baseline approaches to motivate a hop-based correction factor. The hop-based factor balances the extremes of the optimistic and pessimistic factors for guiding the proper link quality correction factor for each user.

Optimistic Correction

The optimistic correction factor makes no changes to the original link quality as determined by the scoped random walk. For all users, the optimistic correction factor is 1:

$$\phi_{opt}(i) = 1, \forall i$$

The optimistic approach will tend to over-estimate the link quality of users that (i) are part of a malicious clique in which some users behave well to mask their relationships with clique members who behave poorly; or (ii) engage in relationships with poor quality users for whom the feedback ratings have incorrectly identified as high quality.

Pessimistic Correction

The pessimistic correction factor treats a user with even a very small likelihood (call it δ) of recommending a poorly performing user as if all of the user's relationship links were to users of feedback rating.

$$\phi_{pess}(i) = \begin{cases} 0 & \text{if } L_{[k]}(i) < 1 - \delta \\ 1 & \text{otherwise} \end{cases}$$

A pessimistic approach may be appropriate in circumstances when relationships with malicious users are highly correlated (as in a malicious clique) or when malicious users in the network are considered extremely dangerous. In this second case, even a single relationship

link to such a dangerous user would warrant a severe correction to the link quality of the recommending user.

Hop-Based Correction

In contrast, the hop-based correction factor seeks to provide a balance between the optimistic and pessimistic correction factors by considering the number and the length of the paths emanating from a user that reach bad users. A *path* in the social network from user i to user j is a sequence of users: $path(i,j) = \langle x_0, x_1, ..., x_n \rangle$ (where $i = x_0$ and $q = x_n$) such that there exists a relationship link between successive nodes in the path, $x_{l+1} \in rel(l)$, for $0 \leq l \leq n-1$. We say a path reaches a bad user if the feedback rating for the user is less than some threshold δ. We call such a path a *bad path*.

For a bad path of length l originating at user i, we associate a hop-based correction factor $\phi_{hop,l}(i)$, where $0 \leq \phi_{hop,l}(i) \leq 1$. By default, we let $\phi_{hop,l}(i) = 1$ if there are no bad paths of length l originating from i. The hop-based discount factor can then be calculated as the product of the constituent discount factors: $\phi_{hop}(i) = \prod_{l=1}^{k} \phi_{hop,l}(i)$.

Selecting the appropriate hop-based correction factor is important, and as we saw in the previous chapter with respect to Web link analysis, there are a number of possible approaches, including a constant, linear, and exponential approach. Due to the success with the exponential-based approach in the Web context, we adopt it here.

Beginning with a user-defined factor ψ ($0 < \psi < 1$) to set the initial hop-based correction for bad paths of length 1, i.e., $\phi_{hop,1}(i) = \psi$, the exponential approach tunes this correction closer to 1 as the bad path length increases:

$$\phi_{hop,l}(i) = 1 - (1 - \psi)\psi^{l-1}$$

meaning that longer bad paths result in a less severe correction to a user's link quality than do shorter paths. Starting with an initial correction factor ψ close to 0 will result in a more pessimistic correction, whereas ψ close to 1 is intuitively more optimistic.

5.7 Assessing the Quality Component of Trust

Recall that the overall SocialTrust trust metric for user i considers both a trust quality component $Tr_q(i)$ and a history component $Tr_h(i)$ (see Equation 10). In this section, we discuss how to compute the quality component of each user's overall trust rating $Tr_q(i)$ (recall Equation 10) through an analysis of the relationship structure in the social network graph $\mathcal{SN}$. In particular, we treat relationships maintained by each user as a list of recommendations of other users, and view the directed graph $\mathcal{SN}$ as a *recommendation-based trust network*, where a relationship link from user i to user j is treated as a recommendation by user i of user j. Based on this recommendation structure, we develop the quality component of trust. The quality component of trust leverages the relationships inherent in the social network to gracefully handle the addition of new users to the network as it evolves. One of the key distinguishing factors of the trust assessment is its incorporation of both feedback ratings and link quality.

5.7.1 Popularity

We begin our development of the SocialTrust quality component by considering a basic trust model that considers the sheer quantity of recommendations for evaluating the trustworthiness of participants. By counting how many recommendations a user receives from other users throughout the network, we can evaluate the popularity of the user in the network. Recall that $rel(i)$ denotes the set of users i has a relationship with (and hence these users have a relationship link to i in the social network graph $\mathcal{SN}$). The popularity-based trust model prescribes a trust value for user i, where:

$$Tr_q(i) = |rel(i)| \tag{11}$$

This recommendation count has close analogs in other network analysis domains, including bibliometrics (where Garfield's Impact Factor measures the importance of academic journals by counting citations) and traditional social network analysis (where popularity can be measured by a count of friends) [129]. The basic popularity trust model is subject to extreme manipulation by malicious (or even just ego-centric) participants. Since online

identities are cheap (often requiring only a valid email address for authentication), malicious cliques can form in which many new users join the network for subverting the trust model; each user in the clique maintains a relationship with a specific target user, resulting in an arbitrarily high popularity and, hence, trust rating for the target user.

5.7.2 Basic Random Walk Model

A natural extension of the basic popularity trust model is to consider both the number of recommendations for a user *and* the quality of the users making the recommendations. Intuitively, if a user is highly trusted, then her recommendations of other users will be considered as more significant than the recommendations of a less trusted user. In this way, the power of malicious cliques can be mitigated to a degree if the colluding users are of sufficiently low quality (and hence their recommendations will count less). This trust propagation formulation can be written in a recursive fashion as:

$$Tr_q(i) = \sum_{j \in rel(i)} Tr_q(j)/|rel(j)| \tag{12}$$

Equivalently, this approach can be described in terms of a random walker who behaves in a manner similar to the random surfer of the popular PageRank approach for Web ranking [138]. The random walker proceeds across the recommendation network; at each node i, the random walker follows one of i's recommendation links with probability $1/|rel(j)|$.[1] In the trust domain, such random walk models have been studied in both the peer-to-peer file-sharing domain [98] and in the context of trust management for the Semantic Web [148]. In the long run, the random walker will visit high-quality users more often than low-quality ones.

5.7.3 Incorporating Link Quality

In Section 5.6, we presented the design of several approaches for evaluating a user's relationship link quality. Naturally, we can incorporate this link quality information into the

[1] For presentation clarity we are assuming that a user's recommendations are treated equally. It is fairly straightforward to generalize to the case of arbitrary recommendation strength; in terms of the random walk, recommendation links will be followed with non-uniform probability according to the recommendation strength.

trust assessment of each user for improving the quality and robustness of the trust assessments. We use the link quality information to impact the size of the vote of each user in the overall trust aggregation process. The intuition is that a user's trustworthiness should be determined by both: (i) the number and trustworthiness of the users who recommend her; and (ii) the link quality of each recommending user. In this way, a recommendation from a high-trust/high-link-quality user counts more than a recommendation from a high-trust/low-link-quality user. Similarly, a recommendation from a low-trust/high-link-quality user counts more than a recommendation from a low-trust/low-link-quality page. Building on Equation 12, the trust score for user i in this link quality enhanced trust model is:

$$Tr_q(i) = \sum_{j \in rel(i)} L(j) * Tr_q(j)/|rel(j)| \tag{13}$$

This formula states that the trustworthiness of user i is determined by the trustworthiness $Tr_q(j)$ and the link quality $L(j)$ of the users that recommend her, as well as by the number of recommendations made by user j (via the factor $|rel(j)|$). In this sense, the recommendation weights are used to determine how a user's "vote" is split among the users that it recommends, but the link quality of a user impacts how large or small is the user's vote.

In practice, the basic random walk formulation must be augmented, both for technical reasons (to insure convergence of the calculation) and for intuitive reasons. The basic random walk can be modified to include a reset probability, such that after walking for a while, the random walker resets to randomly chosen users in the social network. In terms of the random walk, the random walker chooses to follow a user's relationship links (recommendations) with probability λ. With probability $1 - \lambda$, the random walker chooses to reset to another randomly selected user. This resetting approach is useful for ensuring the random walker does not become "stuck" in a clique controlled by malicious users. In the initial case with no additional information about the relative trustworthiness of users, we have a reset probability of $1/n$ for all i. Hence, we can extend Equation 13 to the augmented random walk trust formula:

$$Tr_q(i) = \lambda \sum_{j \in rel(i)} L(j) * Tr_q(j)/|rel(j)| + (1-\lambda)\frac{1}{n} \quad (14)$$

By including the reset probability, however, the random walk model can be made vulnerable to brute-force attacks. A sufficiently large malicious clique can, in essence, attract the random walker via the reset probability $1 - \lambda$ for increasing the trust ratings of its members.

5.7.4 Incorporating Trust Group Feedback

We can further enhance the augmented random walk model by incorporating additional information about each user. Recall that in Section 5.5, we developed several strategies for rating each user through feedback aggregation. These feedback ratings can be used to bias the random walk's reset probability so that it favors users who have been rated highly. These users receive a trust-boost, so that their recommendations have an advantage over users with a low feedback rating. Modifying Equation 13 to include this non-uniform reset probability give us:

$$Tr_q(i) = \lambda \sum_{j \in rel(i)} L(j) * Tr_q(j)/|rel(j)| + (1-\lambda)F(i) \quad (15)$$

Incorporating such feedback information will give an advantage to well-behaved users.

5.7.5 The SocialTrust Model

To complete the trust quality component of SocialTrust, we finally consider the input from a trusted central authority (like the administrators of an online social network) or pre-trusted users. An authority can directly flag malicious users once they are discovered, even if their feedback rating or link quality is high. Similarly, an authority may choose to tune the ratings of users who are new to the online social network that provide some off-line assurances as to their quality (e.g., by submitting to a background check).

If we denote the score assigned to user i by this tunable knob as $A(i)$, then we can enhance the reset probability of the random walk to consider both this authority-based score $A(i)$ and the feedback rating $F(i)$:

$$Tr_q(i) = \lambda \sum_{j \in rel(i)} L(j) * Tr_q(j)/|rel(j)| + (1-\lambda)(\beta F(i) + (1-\beta)A(i)) \qquad (16)$$

where $0 \leq \beta \leq 1$. Choosing $\beta = 0$ favors the central authority's view of user i over any feedback user i may have received. This final SocialTrust trust assessment incorporates the relationship structure of the social network, feedback ratings, link quality, and the tunable input from a trusted central authority. In our evaluation, we shall validate that this approach does, in fact, lead to more tamper-resilient trust ratings than the interim trust models discussed in this section.

5.8 Assessing the History Component of Trust

In the previous sections, we have developed the trust quality component $Tr_q(i)$. We now move to the second key piece of SocialTrust: the history component $Tr_h(i)$ for tracking the evolution of a user's trust rating over time. This history component is important for (i) providing an incentive to all users in the network to behave well over time; and (ii) limiting the ability of malicious participants to whitewash their trust ratings by repeatedly leaving and re-entering the network.

5.8.1 Historical Trust Ratings

The trust quality component $Tr_q(i)$ indicates how well the social network believes that user i can be trusted at point-in-time, but without any consideration of user i's behavior in the past. Suppose the overall trust value $Tr(i)$ is continuously updated. One approach for evaluating the history component of trust considers the integral of the trust value over the lifetime of the user in the network, say, from time 0 to the current time t:

$$Tr_h(i) = \int_0^t Tr(x)dx$$

In practice, the trust computation will be launched at periodic intervals or on a per-need basis, and so this integral will need to be estimated based on discrete trust computations. Assuming that the trust computation is launched at regular intervals for some maximum history H, then the trust manager has access to M historical trust values for user i, denoted

$Tr(i,m)$ for $1 \leq m \leq M$. We can evaluate the history component of trust as a weighted sum over the M historical trust values:

$$Tr_h(i) = \sum_{m=1}^{M} Tr(i,m) \cdot \frac{I_m}{\sum_{l=1}^{M} I_l}$$

where I_m is an importance weight associated with the trust value computed during the m'th time interval. One choice of importance weight is the exponentially weighted sum: $I_m = \xi^m$ for $1 \leq m \leq M$. For $\xi < 1$, the time-averaged trust favors more recent scores for a user over older trust scores. Letting $\xi = 1$ results in a simple average of the trust scores for the history.

5.8.2 Fading Memories

The SocialTrust framework supports the maintenance of history scores for each user up to some system-specified limit M. Choosing a small value for M means that bad behavior will be forgotten by the system after M time intervals. Naturally, we would prefer a large M so that as much historical information about each user is available for trust assessment. Choosing a large M, however, may pose a number of challenges in terms of the space and time burden on the system. Hence, in this section we adopt a *fading memories* approach for balancing the size of the history component with the precision of the historical trust assessments, in a manner similar to the approach proposed in [163].

The goal of fading memories is to summarize the historical trust assessments into intervals of exponentially increasing length, so that more precise trust assessments are available for recent intervals and faded memory estimates are available for older intervals. For tunable parameters f and g, the fading memories approach summarizes $f^g - 1$ trust scores by maintaining just g values, where the intervals are of exponentially increasing length:

$$f^0, f^1, ..., f^{g-1}$$

Suppose $f = 2$ and the current time is t, and the trust manager wishes to maintains a history of $2^g - 1$ trust scores computed at times $t-1$, $t-2$, ..., $t-(2^m-1)$. With fading memories, the trust manager stores only g historical trust scores: one for $t-1$, one for $t-2$

and $t-3$, one for $t-4$, $t-5$, $t-6$, and $t-7$, and so on. For example, for $f=2$ and $g=4$, this means 15 historical trust values may be summarized by storing just 4 values. The 15 values are compressed by keeping scores for intervals of length 1, 2, 4, and 8, rather than the original 15 scores for each interval of length 1. By varying f the system can choose to maintain a longer history of trust scores (with less precision for older scores) or a shorter history of trust scores (with more precision for older scores).

Given the basic fading memory model, it is important to address how the trust manager updates the historical scores as more recent trust scores are added. At each update, there are m available historical scores. Let j be the index of the intervals of exponentially increasing length and $FTr^t[j]$ denote the faded trust value for a user at time t for the j'th interval. Then the faded trust values may be updated at time $t+1$ according to the following formula:

$$FTr^{t+1}[j] = \frac{FTr^t[j] * (2^j - 1) + FTr^t[j-1]}{2^j}$$

Hence, the faded trust values may be updated after each interval using only the g historical trust values, instead of the entire 2^g-1 trust values. For a more detailed treatment of fading memories, we refer the interested reader to [163].

5.9 Personalized Trust Aggregation with mySocialTrust

In the trust framework so far, we have identified feedback ratings and link quality as important features and seen how to assess trust based on each user's position in the social network. We have stressed the overall perspective of the community in assessing trust. In this section, we seek to balance the personal experiences of each user with the larger view of the entire community through a personalized extension to the trust framework called mySocialTrust.

Suppose we are viewing the network from user i's perspective. User i maintains his direct relationships and may have had some direct experience with a small fraction of all community members. In the current model, user i can assess the relative trustworthiness of an unknown user by relying on the community-wide aggregated trust values.

5.9.1 Basic Personalization

One natural approach to provide a more personalized view over these global trust values is to combine the user's direct experiences with the global trust ratings via a linear combination.

$$Tr^{[i]}(j) = \alpha D^{[i]}(j) + (1 - \alpha)Tr(j)$$

where we denote i's direct trust rating of user j by $D^{[i]}(j)$ and the combined trust rating as $Tr^{[i]}(j)$. Regardless of how the personal trust ratings are made (e.g., personal trust ratings may be a function of direct experience or some trust propagation approach), the main drawback of this approach is its sparsity of data for augmenting the trust ratings of most users. Since any one user may have direct experience with only a small fraction of all community members, such a combined trust rating will result in the re-scoring of only a few other nodes.

One way to provide some personalization is to use the distance between nodes to weigh the global trust assignment. If we let $d(i, j)$ denote the shortest distance between two users in a social network (e.g., if user i recommends user k who recommends user j, then $d(i, j) = 2$). Discounting the global trust value of users who are far away in the network leads to the following personalized trust score:

$$Tr^{[i]}(j) = \alpha D^{[i]}(j) + (1 - \alpha)Tr(j)\gamma^{d(i,j)}$$

where γ $(0 \leq \gamma \leq 1)$ is a discount factor for controlling how much to value distant users relative to close users. While such an approach to personalization has the advantage over the basic formulation by considering personalized node distance, it lacks the intuitive appeal of the random walk models advocated in Section 5.7.

5.9.2 Random Walk Personalization

Suppose instead that we augment the original SocialTrust random walk approach described in Equation 16 to consider user i's perspective. Replacing the global values $Tr(j)$, $L(j)$, and $F(j)$ with user i's perspective (respectively $Tr^{[i]}(j)$, $L^{[i]}(j)$, and $F^{[i]}(j)$) we have:

$$Tr^{[i]}(j) = \lambda \sum_{k \in rel(j)} L^{[i]}(k) * Tr^{[i]}(k)/|rel(k)| + (1-\lambda)(\beta F^{[i]}(j) + (1-\beta)A(j)) \quad (17)$$

We can interpret $Tr^{[i]}(j)$ as user i's trust rating of user j. But how are the personalized link quality and feedback ratings made in the first place? Since the trust rating is driven by these two inputs, it is of critical importance to identify how they are made.

One approach uses a similar spirit of the hop-based trust dampening suggested above to influence the link quality and feedback ratings for each user. In this case we can replace $L^{[i]}(k)$ and $F^{[i]}(j)$ with estimates based on the global values and the distance of each user j from user m:

$$L^{[i]}(k) = L(k)\gamma^{d(i,k)}$$

$$F^{[i]}(k) = F(k)\gamma^{d(i,k)}$$

Plugging these values into Equation 17 yields a recursive formulation in which the trust value of each user is impacted by the relative link quality and feedback rating as a function of the user's distance from the point of view of user i. This formulation has the nice property of balancing the local and global perspectives relative to i. A user that is farther from i will require proportionately more high quality recommendations (via the link quality factor) to score as high as a user much closer to i. In this way, users that are deemed globally of high link quality and trust can score high even if they are quite distant from user i.

5.10 Experimental Setup

We evaluate the SocialTrust framework through a series of simulations over real social network data. In this section, we describe the simulation setup for our evaluation of the SocialTrust framework. We present the data, experimental setup, and metrics for evaluation. Note that in our evaluation we focus exclusively on the quality component of trust. For further discussion of trust history and its impact on trust effectiveness, we refer the interested reader to [163].

5.10.1 Data

All of the experiments reported in this chapter rely on data collected from the MySpace social networking site. MySpace is currently the largest social networking site and one of the few that provides open access to public user profiles. Many other sites (e.g., Facebook, LinkedIn) require a user account and, even then, access to the entire social network can be restricted. We ran multiple parallel crawlers over MySpace in July 2006, beginning from a random sample of seed profiles. The crawlers followed the relationship links listed on each profile's front page in a breadth-first traversal of MySpace, resulting in a collection of 891,197 full-text profiles. Based on these profiles, we generated a directed social network graph consisting of 5,199,886 nodes representing both the collected full-text profiles and additional referenced profiles and 19,145,842 relationship links. As a pre-processing step, we removed the default relationship links to the profile belonging to "Tom", the purported creator of MySpace who serves as a default friend for all new users, and whose super-user presence is atypical of most social networks.

We performed validation of the graph, in which we confirmed that the link distribution of the graph follows a power-law and that the clustering coefficient (an indicator of the small-worldness of graphs) was 0.06. Both features have been observed to be important in social networks [103, 175]. We also confirmed that the graph is connected.

5.10.2 Setup

The simulation begins from a cold start, in which each user in the network is assigned a default trust score. Thereafter, users issue queries to the network according to the *forwarding-based search capability*, report their feedback to the trust manager, and at regular intervals the trust manager calculates the trust score for each user in the network for use in the next cycle. All trust calculations are performed using the Jacobi method for 25 iterations and a mixing parameter $\lambda = 0.85$. In all of our experiments, a simulation cycle consists of 5,000 queries. There are 30 simulation cycles in total. For each query, users provide feedback over the top-20 responses received. We report results over the last 5,000 queries, averaged over five simulation runs.

Queries are randomly selected from a query dictionary composed of all the terms in the collected MySpace profiles. Each candidate query q is weighted by the number of profiles in which it occurs. When a user receives a query, she performs a simple binary match between the query term and her profile terms. If the profile contains the term, there is a match, and the user responds to the query originator. In our threat scenarios, we shall consider two types of users: (i) malicious users, who always provide an irrelevant response to queries that they receive; and (ii) legitimate users, who sometimes accidentally provide an irrelevant response to queries that they receive.

5.10.3 Metrics

In this section, we discuss several metrics for measuring the cost and quality of search and ranking in online social networks.

5.10.3.1 Cost

Depending on the search strategy, a query may be forwarded to many users in the social network or very few. We measure the cost of search in terms of how loaded the system becomes in terms of users contacted per query and messages sent per query:

- **Users Contacted:** We measure the cost of search in terms of how many users are contacted per query. When a query reaches a user, the user can choose to respond directly to the originating user or can choose to pass the message along to some subset of her friends. A query that contacts more users requires either more user attention or more system resources for forwarding and query processing.

- **Number of Messages:** Related to the first cost metric is the number of messages passed from user to user for satisfying a query. A duplicate message could arrive at a user through different relationship chains. A search strategy that communicates a query to a user only once is preferred to a strategy that communicates with the same user multiple times for the same query.

5.10.3.2 Quality of Results

The goal of social search and ranking is to identify high-quality results for a query, even in the face of intentional user manipulation. For a query q, let R^+ denote the set of relevant users for q throughout the entire space of users, let R denote the set of all responses to q, and let R_n denote the n-highest-ranked responses. We measure the quality of search and ranking through two commonly used information retrieval metrics.

- **Precision at N:** Since a user may be only interested in the top-ranked results returned, we measure a focused version of the standard precision measure that considers the quality of the results returned in the top-n (e.g., for $n = 1, 5, 10$): $prec_n = \frac{|R^+ \cap R_n|}{n}$
- **Relative Precision at N:** The precision@n metric penalizes a ranking in which fewer than n results are returned, even if the returned results are all relevant. Hence, we also measure the relative precision@k that measures the quality of results only over the results returned, even if fewer than n are returned: $\widehat{prec}_n = \frac{|R^+ \cap R_n|}{min(|R_k|,n)}$

Note that the traditional precision measure over all candidate results – $precision = \frac{|R^+ \cap R|}{|R|}$ – provides little distinguishing power since malicious users may overwhelm an originating user with many poor quality results, even if the top-k are of high-quality. The recall measure – $recall = \frac{|R^+ \cap R|}{|rel|}$ – also provides little distinguishing power among ranking strategies since any search is expected to identify only a fraction of the possible relevant results.

For each quality metric, we measure the average performance over many queries issued to the network from many different originating users, so that we can identify system-wide quality metrics for comparing trust models.

5.10.4 Baseline Trust Model

In all of the reported experiments, we use the SocialTrust trust model described in Equation 10 using the quality component described in Equation 16. For the link quality component, we rely on the scoped random walk model with scope of $k = 3$ and an exponential correction factor with $\psi = 0.5$ and $\delta = 0.5$. For now, we shall ignore the history

component and assume no trusted central authority for making corrections via the tunable knob $A(i)$. We rely on a relationship-based trust group with radius 7 for which we shall search using random selection with up to eight random neighbors at a horizon of 7. We shall revisit some of these assumptions in the following experiments.

5.11 Evaluation

In this section, we evaluate the SocialTrust framework. through a series of experiments over real social network data collected from MySpace. We focus on four aspects: (i) a comparison of the trust group search strategies; (ii) a comparison of SocialTrust versus alternative trust models; (iii) the study of link quality; and (iv) an evaluation of SocialTrust in the presence of strategies attempting to subvert its effectiveness, including clique formation and collusive feedback. We find that the SocialTrust framework supports robust and tamper-resilient trust ratings even when large portions of the social network engage in behavior intended to undermine its effectiveness.

5.11.1 Comparing Trust Group Search Strategies

In Section 5.4, we described the formation of relationship-based trust groups for each user. A user's trust group is governed by the network-specified trust group radius. When a user queries the social network, she forwards the query to some portion of the trust group. In this first set of experiments, we explore the relative quality of several search strategies over these trust groups as a baseline for the rest of our evaluation of the SocialTrust framework.

Given a fixed ranking model, a search model that explores very little of the social network may provide high quality responses for the queries that it can answer, but overall provide little value if it can answer only a small fraction of all queries. On the other hand, a search model that explores a wide range of the social network may be able to satisfy many queries, but at the cost of burdening the social network by contacting too many users.

We randomly identify 25% of the users in the network as malicious. When contacted, these malicious provide a corrupt result with 100% probability. Since regular users may sometimes accidentally provide incorrect or corrupt results, all other users respond with a corrupt result with 5% probability.

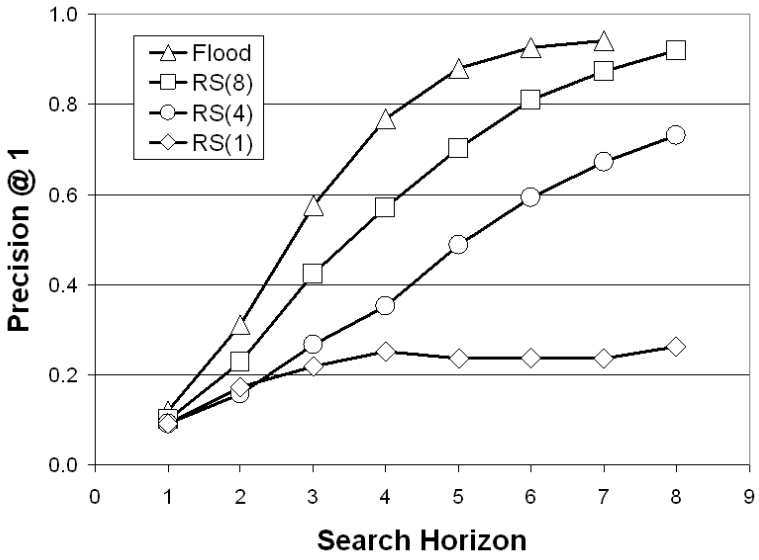

Figure 51: Trust Group Search Strategy: Precision @ 1

In Figures 51 to 55, we compare the flooding-based search strategy (*Flood*) versus three random selection strategies that select one neighbor $RS(1)$, up to four neighbors $RS(4)$, and up to eight random neighbors $RS(8)$. In Figure 51, we evaluate the precision @ 1 for each of the four strategies for an increasing search horizon. In all cases as the search horizon increases, more of the social network is explored and the precision for all strategies increases, except the $RS(1)$ strategy. The $RS(1)$ is a depth-first exploration of the network, contacting very few users in the network.

Naturally, *Flood* and $RS(8)$ perform the best, but it is interesting to note that the precision in both cases approaches 90% for a large search horizon. Even in the presence of 25% malicious users, this indicates that the SocialTrust framework supports high-quality tamper-resilient rankings. Note that the precision does not reach 100% since even non-malicious users will sometimes answer a query with a non-relevant response.

Coupled with this result, we show in Figure 52 the precision @ 10 for the same four search strategies. As in the precision @ 1 case, the *Flood* and $RS(8)$ strategies perform the best, and we note only a slight drop in precision across all cases.

In Figure 53, we report the relative precision @ 10 for the four search strategies over only the queries that were actually answered. Here we see that even for the search models

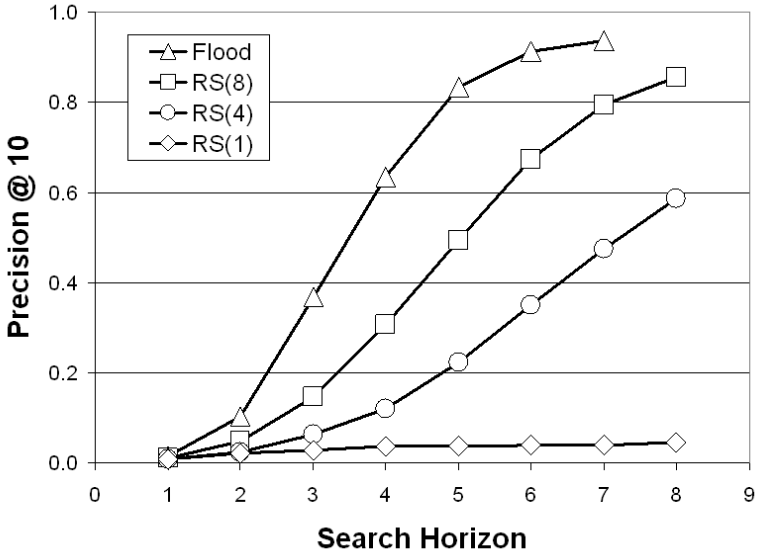

Figure 52: Trust Group Search Strategy: Precision @ 10

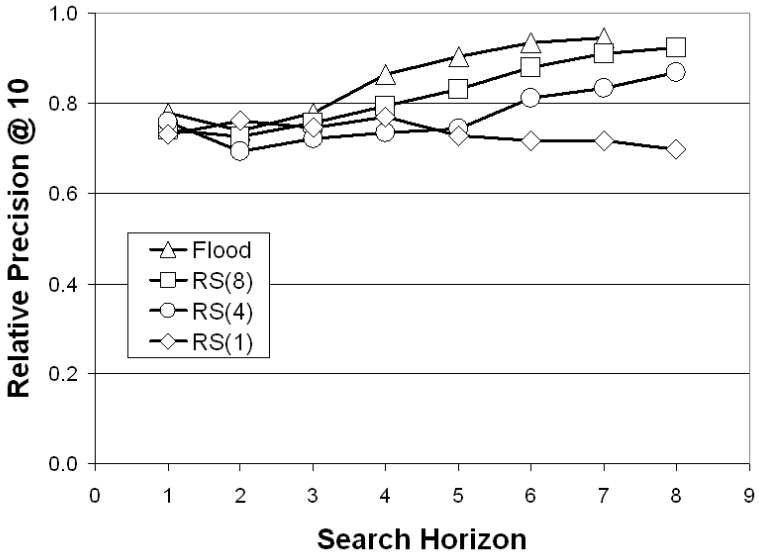

Figure 53: Trust Group Search Strategy: Relative Precision @ 10

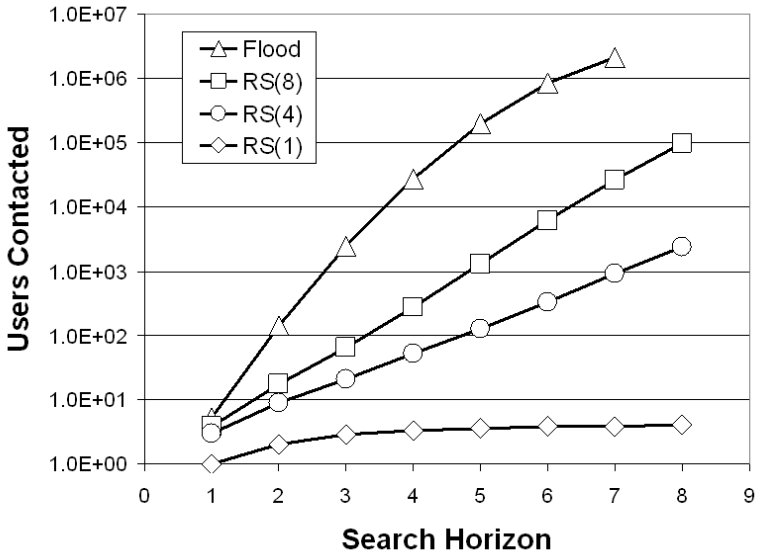

Figure 54: Trust Group Search Strategy: Users Contacted

that answer only a small fraction of all queries that they do provide high quality responses for the queries that they do answer. This suggests that the SocialTrust framework is providing high-quality responses and that it is primarily the search strategy over the trust group that is limiting the precision.

We compare the cost of the four search strategies by measuring the number of users contacted by each, and by the message premium – that is, the Number of Messages / Users Contacted. In Figure 54, we see that the users contacted per search strategy grows exponentially (note that the y-axis is log-based). At a horizon of 7 the flooding strategy contacts nearly half of the entire social network with a message premium of 200% (see Figure 55), meaning that each user is being contacted twice per query issued to the network.

Finally, we consider the impact of users making a more informed decision when deciding to whom to forward a query. In practice, users will have some knowledge of which neighbors are more suited to answer a query. We compare the random selection strategy versus the selective forwarding search strategy, where a user relies on a fuzzy match function between the profile published by each neighbor versus the query. For each buddy $j \in rel(i)$, we assess the neighbor match function $NeighborMatch(q, j)$ based on a Gaussian distribution with noise, such that $NeighborMatch(q, j) \sim N(\mu + \delta, \sigma^2)$ if user j's profile contains the

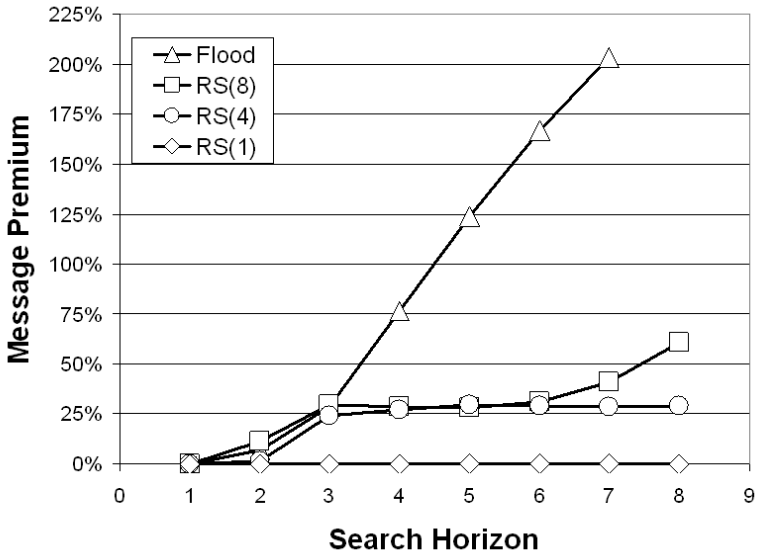

Figure 55: Trust Group Search Strategy: Message Premium

query term q. If not, then the match is estimated as $NeighborMatch(q, j) \sim N(\mu - \delta, \sigma^2)$. We consider $\mu = 0$ and $\sigma^2 = 1$, and compare three version of selective forwarding based on $\delta = 0.1$, $\delta = 1$, and $\delta = 2$. A higher δ means that a user is more confident in assessing the quality of his neighbors for answering a query.

In Figure 56, we compare random selection versus the three versions of selective forwarding – $SF(0.1)$, $SF(1)$, and $SF(2)$ – for a horizon of 7 and maximum neighbors selected by each user of 8. We see that in all cases selective forwarding does result in a slightly higher precision.

In the rest of the evaluation reported here, we shall rely on a trust group with radius 7 for which we shall search using Random Selection with up to eight random neighbors ($RS(8)$) at a horizon of 7. This search strategy contacts orders of magnitude fewer users per query with a much less message premium ($\sim 40\%$) versus the flooding strategy, and it achieves a precision comparable to the flooding strategy. Since this strategy balances the cost on the network with high precision results, we shall use it as our standard search strategy for the rest of our reported experimental results.

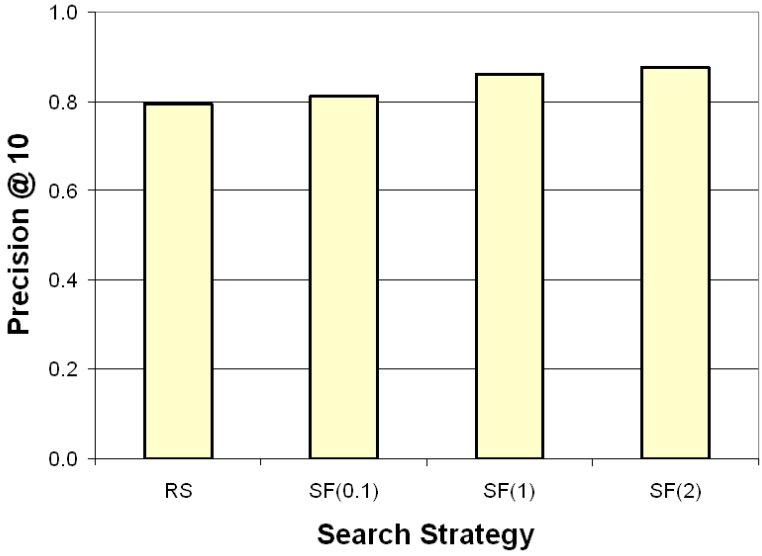

Figure 56: Trust Group Search Strategy: Informed Search

5.11.2 Comparing Trust Models

In the next set of experiments, we evaluate the quality of SocialTrust against alternative trust models and for varying degrees of user manipulation within the network. We again use the SocialTrust trust model described in Section 5.10.4.

We first compare SocialTrust against the *No Trust* case – in which a user randomly selects among the returned query responses – and a simple trust model based on the *Popularity* of the user in the network (refer to Equation 11). For each of the trust models, we consider seven scenarios: 10% of the network is malicious, 20%, ..., 70%. Malicious users provide a corrupt result with 100% probability; other users respond with a corrupt result with 5% probability. In all cases, if 100% of the network is malicious, the trust ratings are meaningless and the overall search precision drops to 0.

In Figure 57 and Figure 58, we report the relative precision @ 10 and the relative precision @ 5 for each of these trust models and scenarios. In both cases, we see that the relative precision for SocialTrust is resilient to the increase in malicious users, whereas the *No Trust* and *Popularity* models degrade severely.

The rapid degradation in precision quality for the *No Trust* model is not surprising. With an increasing number of malicious users in the network, the *No Trust* model gives the

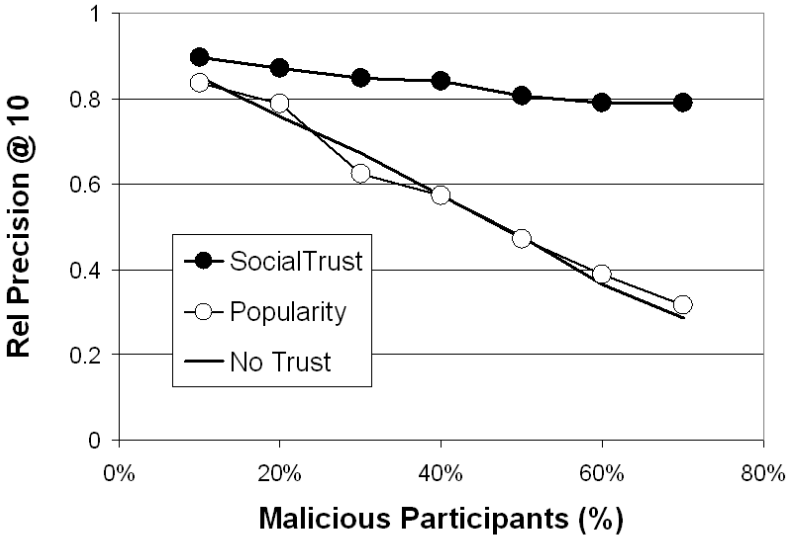

Figure 57: Trust Models: Relative Precision @ 10

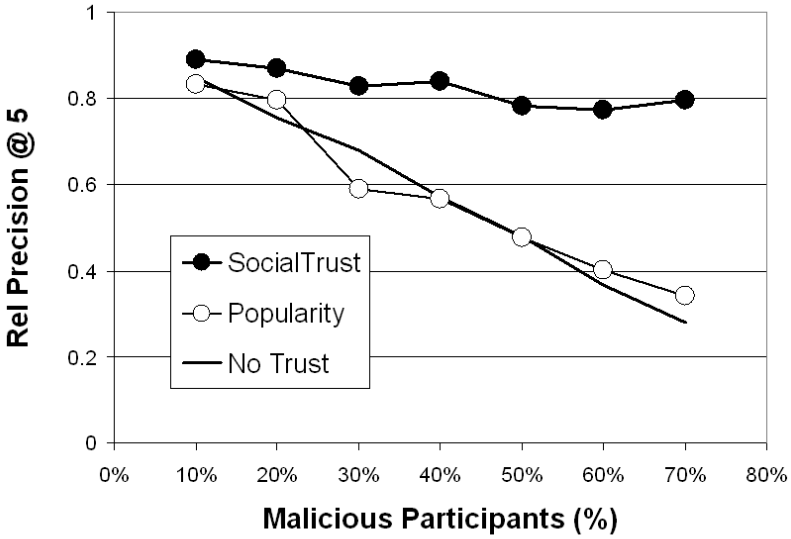

Figure 58: Trust Models: Relative Precision @ 5

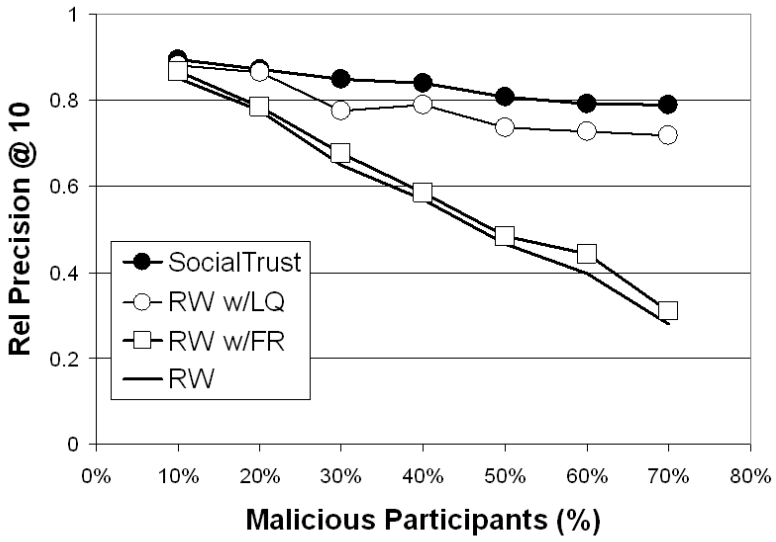

Figure 59: Comparing Random Walk Trust Models: Relative Precision @ 10

unsuspecting user who queries the network no assurances as to the quality of the returned results. At first glance, the fall in precision for the *Popularity* model may be surprising, but consider that malicious users are distributed throughout the network at all ranges of popularity. Hence, when a proportion of the most popular (and most trusted) users behave maliciously, there is no mechanism for correcting this bad behavior.

In contrast, the SocialTrust model incorporates link quality and feedback ratings into the trust assessment so that bad behavior is punished, and so the resulting precision measures are resilient to the presence of a large fraction of malicious users in the network. This is especially encouraging since the feedback ratings available in one simulation round may be incomplete for users who have not yet responded to a query in previous rounds.

To further illustrate why the SocialTrust model performs so well, we next consider several related trust models described in Section 5.7. We consider a simple random walk model (RW) like the one in Equation 12 that considers only the relationship structure of the social network; a random walk model that incorporates feedback ratings ($RW\ w/FR$) like the one in Equation 15 but without link quality; and a random walk model that incorporates link quality but without feedback ($RW\ w/LQ$) as in Equation 14.

In Figure 59 and Figure 60 we report the relative precision @ 10 and the relative precision

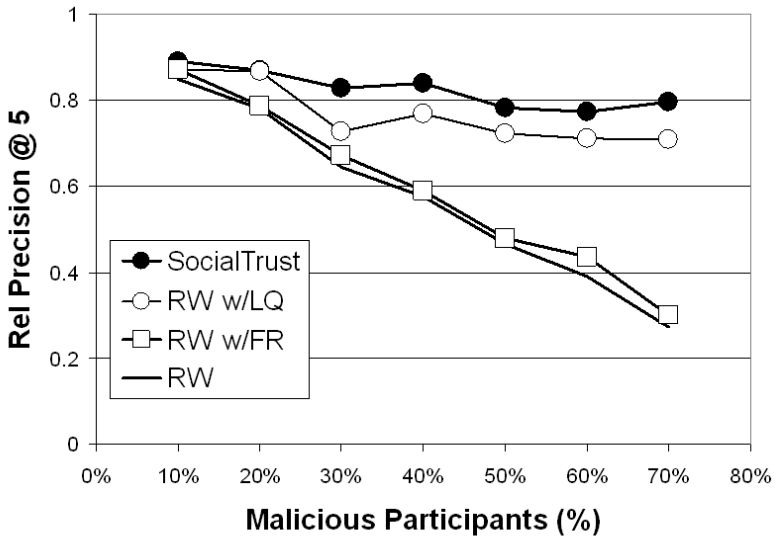

Figure 60: Comparing Random Walk Trust Models: Relative Precision @ 5

@ 5 for an increasing proportion of malicious users for each of these trust models. Note that the basic random walk model performs approximately the same as the *No Trust* and *Popularity* models shown in Figure 57. Incorporating feedback provides only a slight boost in the precision. In contrast, we see that incorporating link quality provides the single biggest improvement in precision, since it reduces the influence of users who engage in poor quality relationships. When coupled together, both feedback ratings and link quality provide the overall best performance.

To further illustrate, we show in Figure 61 the relative precision @ 10 for the scenario when 50% of the network is malicious.

5.11.3 Impact of Link Quality

Since link quality is such an important factor, we next compare several versions. We additionally the optimistic approach for $k = 1$ to $k = 5$, the pessimistic approach for $k = 1$ to $k = 5$ (with $\delta = 0.5$), as well as the exponential hop-based approach for $k = 1$ to $k = 5$. In Figure 62, we report the relative precision @ 10 for the scenario when 50% of the users in the network are malicious, but with the different approaches for computing link quality incorporated into the trust model.

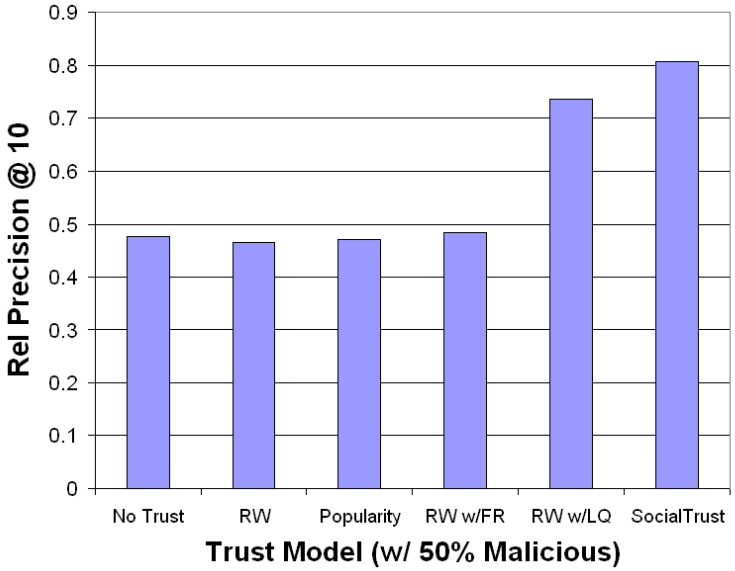

Figure 61: Comparing Random Walk Models: 50% Malicious

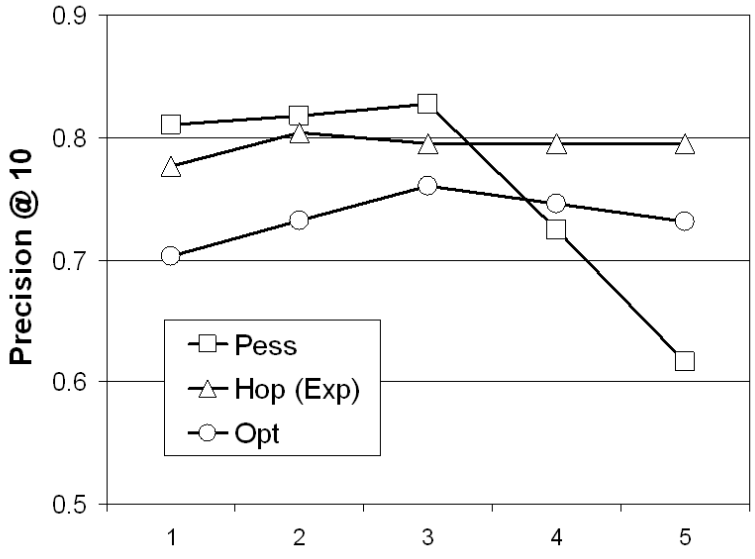

Figure 62: Comparing Link Quality Approaches

First, the optimistic and hop-based approach are stable and perform fairly well as the scope parameter k increases. These approaches penalize a candidate user's link quality score in proportion to the distance of malicious users from the candidate user. Direct links to malicious users result in a lower link quality score than paths of multiple hops to malicious users. In contrast, the pessimistic approach results in a worsening of precision as the scope increases. As k increases, most users have at least one path to a malicious user and are assigned a 0 or low link quality score. As the link quality score approaches 0 for nearly all users in the network, the rankings induced from the trust model (recall Equation 10) become essentially random, and so we see the precision fall considerably.

5.11.4 Clique Formation

We have witnessed how the SocialTrust approach provides resilient ranking support for search in online social networks. We next consider several alternative scenarios and evaluate the robustness of the approach. In our previous experiments, malicious nodes enter the network randomly. Suppose instead that malicious nodes seek to form cliques in the social network so that they can leverage their tightly-coupled relationship structure to overpower the quality component of trust (recall Section 5.7).

For this experiment, rather than randomly assigning users to be malicious, we now construct cliques of malicious users. The setup works like this: first a node is randomly selected and assigned to be a malicious node, then up to three-hops of its neighbors are also assigned to be malicious. We repeat this process until 10% of the network is malicious. This overall procedure continues for the 20% case, 30% case, up to the 70% case.

In Figure 63 we report the relative precision @ 10 for SocialTrust over this clique-based strategy (*Clique*). As points of comparison, we also show the performance of SocialTrust over the original non-clique strategy (*Non-clique*), as well as the performance of the *No Trust* strategy over the clique-based strategy. Even in the presence of cliques, the SocialTrust approach provides resilient rankings as the fraction of malicious users increases. For many malicious users, the non-clique case behaves in a manner similar to the clique case. With so many malicious users, by chance there will be many random cliques

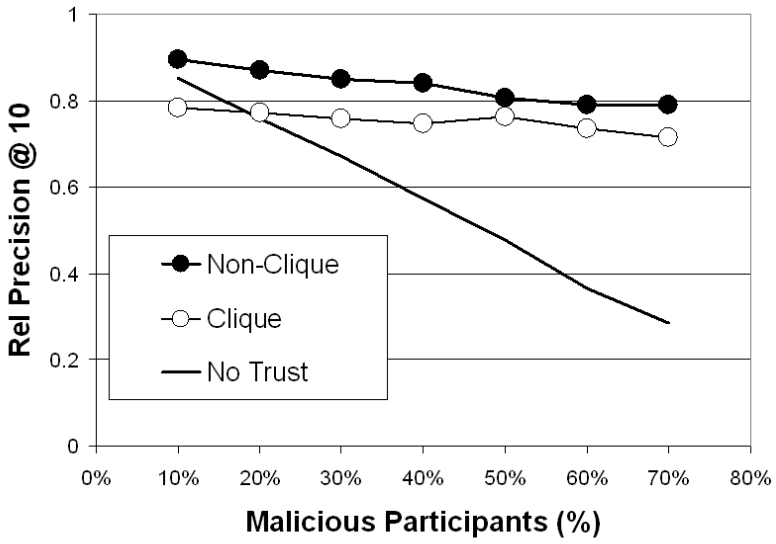

Figure 63: Effectiveness of Clique Strategies

formed. We attribute the success of the SocialTrust approach to its incorporation of link quality, so that the influence of malicious cliques over the aggregated trust ratings is reduced.

5.11.5 Subverting Feedback

Suppose that in addition to providing poor search results, that malicious users also attempt to subvert the feedback ratings. So far, we have used the Trust-Aware Restricted Voting at the end of each simulation cycle, where a user's feedback is proportional to his trust rating. In this final experiment, we consider the other two voting schemes discussed in Section 5.5 – open voting and restricted voting.

Recall that the Trust-Aware approach allots a voting share to each user based on his trust value, so that more trusted users have greater sway over the feedback ratings of other users than do lowly trusted users. For the restricted voting case, each user is allotted an equal voting share for distributing among the users in his trust group who have answered its queries in the past. In the open voting case, there are no constraints on the number of votes cast by any user.

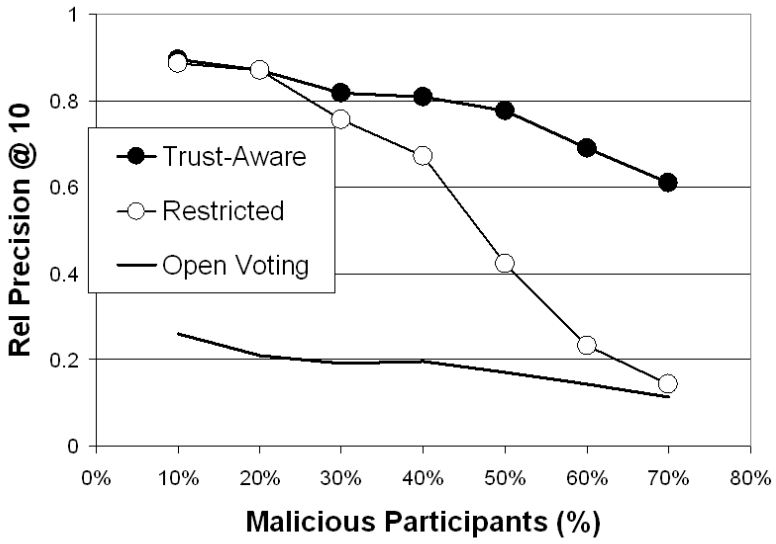

Figure 64: Comparing Feedback Schemes

For each voting scheme, we assume that a malicious user always provides negative feedback for legitimate users, regardless of the quality of the search result provided; a legitimate user provides honest feedback. For the open voting case, we assume that malicious users ballot stuff the voting process, resulting in feedback ratings for legitimate users randomly distributed between $[0, 0.1]$. Malicious users receive high feedback ratings randomly distributed between $[0.9, 1]$.

In Figure 64, we compare the performance of the SocialTrust framework over each voting scheme. As the network tips over 50% malicious, the restricted voting case begins a steep decline. In this scenario, there are more malicious users in the network and so (regardless of their past trust values), they can promote other malicious users, so that in the following round these malicious users receive a boost in feedback rating (and hence, link quality, and ultimately, trust). For the open voting scheme, we see that precision is very low across the scenarios. In this case, even a small percentage of malicious nodes can subvert the feedback ratings of legitimate users (and promote the scores of other malicious users), so that the derived trust ratings favor malicious users.

In contrast, the trust-aware voting scheme is fairly resilient; as more and more malicious users enter the network, the highly-trusted users manage to keep them under control. The

robustness of the SocialTrust model, even with large portions of the network providing dishonest feedback, can be partially attributed to our model of how malicious users enter the network. In our simulations, malicious user are activated in 10% chunks. Since trust and feedback ratings are linked from round-to-round, the votes of legitimate users in one round can deter the malicious users from receiving high trust scores in the following round. In contrast, if 70% of the entire network were to suddenly behave maliciously, we would observe a steep degradation in precision. This experiment emphasizes the need for high-quality feedback aggregation; without it, even the best trust model is subject to extreme manipulation.

5.12 Related Work

The study of social networks has a rich history [126], and there are a number of tools for the analysis of social networks [174]. Recently there have been some advances in studying small-world networks [175, 176] and how to search traditional off-line social networks [54]. The rise of online communities has spurred interest in community information management [53], social network formation [11], and the modeling of online social networks [103, 108]. One recent study examined email networks for analyzing how users navigate an online social network [3].

The advances made in peer-to-peer networks, e.g., [146, 164], could inform the development of robust and scalable online social networks. For example, there have been a number of advances in efficiently searching unstructured P2P networks, as in [47], [116], [188], and [187]. Note that there are some key differences which could prove important for building effective and efficient search mechanisms for online social networks. P2P networks often are concerned with high node churn and guaranteeing anonymity; the networks are often formed via randomization protocols for establishing links between nodes. In contrast, online social networks tend to include long-lived profiles that strive to be known (i.e., are not anonymous); links in the social network stress the personal connection. There have been several attempts in the P2P domain to manage trust and reputation, including [1], [45], and [48].

Trust and reputation are important computational concepts. For an overview of trust, we refer the interested reader to [119]. Several studies have examined how to compute direct trust between nodes, including: [152], which developed statistical models of bid behavior on eBay to determine which sellers are suspicious; TrustGuard [163], which targeted strategic malicious nodes who oscillate their behavior; PeerTrust[183], which studied feedback credibility; and [120], which stresses the need for direct trust. How to propagate trust through a network was studied in [75], where the notion of distrust was also considered. Yu and Singh suggested a framework for reputation formation in electronic communities [189].

5.13 Summary

In this chapter, we have presented the design and evaluation of the SocialTrust framework for aggregating trust in online social networks. The framework supports tamper-resilient trust establishment in the presence of large-scale manipulation by malicious users, clique formation, and dishonest feedback. We have introduced relationship link quality as a scoped random walk and explored key factors impacting it. Link quality can be optimized depending on the context and application scenario and on the risk tolerance of the social network. We have developed a new random walk trust model incorporating relationship link quality and trust group feedback, and provided the first large-scale trust evaluation over real social network data.

CHAPTER VI

CONTROLLED SAMPLING OF WEB RESOURCES

In this and the following chapter, we address the third and final major thesis contribution – tamper-resilience in Web categorization and integration services. These services are important for organizing and making accessible the large body of Web-enabled databases on the *Deep Web* that are beyond the reach of traditional Web search engines. Recent estimates report the size of the Deep Web at nearly 92,000 terabytes of data versus only 167 terabytes on the surface Web [117] of Web documents that reside physically in a file directory reachable by the respective Web server. Deep Web data resources are growing fast. The practical size of Deep Web resources is over 450,000 autonomous databases, offering over 1.2 million unique search services. The number of resources more than quadrupled between 2000 and 2004 [39], and considerable data on the Deep Web is subject to real-time changes [39].

Most existing Web crawlers tend to ignore the data offered by Deep Web databases due to the technical difficulties of locating, accessing, and indexing Deep Web data. As a result, a number of services have been proposed to support categorization of these information resources as well as integration services for providing uniform query-access to these resources, e.g., [74, 86, 93, 150, 182].

To reduce the opportunities of attackers to corrupt Web-based categorization and integration services, we introduce in this chapter a controlled sampling architecture for extracting representative summaries of each Web resource, in an effort to avoid the possibly corrupt or untrustworthy self-published metadata and categorization information.

6.1 Controlled Sampling Overview

To enable advanced information services (such as classification and integration) over the large and growing number of distributed information resources like those on the Deep Web, traditional approaches for generating summary information for each text database under consideration through crawling *all* available content are unrealistic for a couple of reasons.

First, most Deep Web resources are autonomous and offer limited query capability through a site-specific query interface, making it difficult if not impossible to obtain the complete collection of their data through query probing, especially for resources that are unwilling to allow complete access to their entire archives. Second, the cost of collecting all Deep Web data from a large number of Deep Web resources is prohibitive due to the computational burden on each Deep Web resource to answer every query. As a result, *query-based sampling* mechanisms have become a popular approach for collecting document samples from each Deep Web resource of interest.

By query-based sampling from each Deep Web resource, we may support a host of advanced information services built on the sampled data, including: (i) the automated categorization of Deep Web resources for use by a Yahoo!-style Web directory service; (ii) the clustering of Deep Web resources for discovering interesting relationships among Deep Web resources; (iii) the routing of queries so that user queries are only sent to the most appropriate Deep Web resources for evaluation; and (iv) the augmenting of traditional search engines, so that Deep Web data may be indexed and made available alongside surface data that has been crawled by the search engine. The sampling-based approach has garnered previous research attention and shown success in several contexts, including distributed information retrieval, database selection, database categorization, peer-to-peer information retrieval, to name a few [35, 91, 114, 160].

There are three key challenges for using a sampling-based approach to analyze distributed text databases and Deep Web resources. First, a sampling-based approach must understand the query interface of each distributed text database for effectively sampling from the space of all documents available at the database [*how to sample*]. This entails parsing the query interface, automatically filling out the query interface, and selecting appropriate queries to probe the target databases. Second, given the rate of change and evolution of distributed data resources, we need to consider the appropriate sampling schedule to maintain sample freshness [*when to sample*]. Finally, given realistic network constraints and the size and growth rate of distributed text databases, we require an approach for knowing what to sample from each database [*what to sample*]. Important questions related

to *what to sample* include (i) should each resource be treated equally? (ii) should we devote a higher sampling allocation to certain databases to obtain high-quality samples? (iii) how do we determine the quality of each sample? and can we use a dynamic quality-conscious approach to manage the overall sampling process?

While there have been some previous efforts to study how to sample (e.g., [37, 145]) and when to schedule such samples (e.g., [92]), there has been no dedicated study on the problem of *what to sample* from each text database given a collection of databases and a resource constraint for sampling. In this chapter we provide the first comprehensive investigation of the problem of *what to sample* from each text database. Concretely, given a set of n distributed databases, each of which provides access primarily through a query-based mechanism (like a keyword search interface), and given limited resources for sampling all n databases, can we identify an optimal strategy for extracting the highest-quality samples from the n databases? We refer to this problem as the *distributed query-sampling problem*.

In this chapter, we present an *adaptive distributed query-sampling framework* that is quality-conscious for optimizing the quality of database samples. The framework divides the query-based sampling process into initial seed sampling phase and a quality-aware iterative sampling phase. In the second phase the sampling process is dynamically scheduled based on estimated database size and quality parameters derived *during* the previous sampling process. The unique characteristic of our adaptive query-based sampling framework is its self-learning and self-configuring ability based on the overall quality of all text databases under consideration. We introduce three quality-conscious sampling schemes for estimating database quality, and evaluate our adaptive sampling approach using standard TREC information retrieval data collections. Our results show that the adaptive query-based sampling framework supports higher-quality document sampling than existing approaches. We also evaluate our adaptive sampling framework using the real-world application of database selection, demonstrating the quality and performance enhancement of our approach.

The remainder of the chapter is organized as follows. Section 6.2 describes how text databases are modeled and the basics of metadata acquisition through sampling. We then introduce the adaptive architecture for distributed query sampling in Section 6.3, including

three quality-conscious sampling schemes. We present the data and metrics in Section 6.4 along with a series of experiments for evaluating the quality of the sampling framework. We then conclude the chapter with related work in Section 6.5 and a summary in Section 6.6.

6.2 Basic Reference Model

Before introducing the distributed query-sampling framework, we first describe the basic reference model used in this chapter, including how text databases are modeled, how document samples are extracted from each text database using query-based sampling, and the representative state-of-the-art solution for the distributed text database sampling problem.

6.2.1 Modeling Text Databases

We consider a *text database* to be a database that is composed primarily of text documents and that provides query-based access to these documents either through keyword search or more advanced search operators. Examples of text databases include Deep Web data resources, digital libraries, and legacy databases searchable only through query-based mechanisms.

We consider a universe of discourse $\mathcal{U}$ consisting of n text databases: $\mathcal{U} = \{D_1, D_2, \ldots, D_n\}$ where each database produces a set of documents in response to a query. A text database D is described by the set of documents it contains: $D = \{doc_1, doc_2, \cdots\}$. We denote the number of documents in D as $|D|$. We denote the set of terms in D as the *vocabulary* V of D. The number of unique terms, denoted by $|V|$, is referred to as the *vocabulary size* of the database D. We also track the following detailed term statistics: $c(t, D)$ denotes the count of documents in D each term t occurs in; and $f(t, D)$ denotes the frequency of occurrence of each term t across all documents in D (i.e., how many times the term appears in the text database).

A document *sample* from database D, denoted by D_s, consists of a set of documents from D. We use $|D_s|$ to denote the number of documents in the sample D_s, where typically $|D_s| \ll |D|$. We refer to the set of terms in D_s as the *sample vocabulary* V_s of D_s. The number of unique terms $|V_s|$ is referred to as the *sample vocabulary size* of the database sample. Similar to the text databases, we also track the following detailed term statistics for

Table 9: Text Database Notation

D	a text database, composed of documents
V	vocabulary of D
$\|D\|$	number of documents in database D
$\|V\|$	size of vocabulary V
$c(t, D)$	count of documents in D each term t occurs in
$f(t, D)$	frequency of occurrence of each term t across all documents in D
D_s	sampled set of documents from database D
V_s	vocabulary of D_s
$\|D_s\|$	number of documents in sample D_s
$\|V_s\|$	size of vocabulary V_s
$c(t, D_s)$	count of documents in D_s each term t occurs in
$f(t, D_s)$	frequency of occurrence of each term t across the sampled documents D_s
$f(t, doc, D_s)$	frequency of occurrence of term t in document *doc* in D_s

each sample D_s: $c(t, D_s)$ denotes the count of documents in D_s each term t occurs in; and $f(t, D_s)$ denotes the frequency of occurrence of each term t across the sampled documents in D_s. We additionally track $f(t, doc, D_s)$ = the number of occurrences of a term t for each *doc* in D_s. Figure 9 summarizes the notation introduced in this section for ease of reference.

6.2.2 Query-Based Sampling

To support database sampling, Callan and his colleagues [36, 37] have previously introduced the query-based sampling approach for generating estimates of text databases by examining only a fraction of the total documents. The query-based sampling algorithm works by repeatedly sending one-term keyword queries to a text database and extracting the response documents:

Steps of Query-Based Sampling from a Database D

1: Initialize a query dictionary Q.

2: Select a one-term query from q from Q.

3: Issue the query q to the database D.

4: Retrieve the top-m documents from D in response to q.

5: [Optional] Update Q with the terms in the retrieved documents.

6: Goto Step 2, until a stopping condition is met.

Critical factors impacting the performance of Query-Based Sampling algorithms include the choice of the query dictionary Q, the query selection algorithm, and the stopping condition. The optional step 5 allows the algorithm to learn a database-specific query dictionary as a source of queries after the first successful query from the initial query dictionary. Research [36, 37] has shown that document samples extracted by Query-Based Sampling can be of high quality based on a number of quality metrics. Concretely, these studies show that document samples consisting of a fairly small number of documents (e.g., 300) are of high quality over text databases consisting of millions of unique documents. Since the distribution of terms across documents in a text database typically follows a Zipfian distribution – meaning a few terms occur across many documents, but most terms occur in only a few documents – sampling a small number of documents can extract many of the high-frequency terms in the text database. As a result, the sample vocabulary will cover only a fraction of all possible terms in the database (that is, $|V_s| \ll |V|$).

6.2.3 Sampling From Distributed Text Databases

The problem of distributed query sampling is to determine what to sample from each of the n text databases under a given sampling resource constraint. To simplify the discussion without loss of generality, we assume that there are uniform sampling costs from each database, and that the distributed query sampling algorithm may sample at most a total of S documents from the n databases. Hence, the goal of the distributed query-sampling framework is to identify the optimal allocation of the S documents to the n databases. An effective sampling framework must address the competing issues: the quality of each database sample versus the overall quality of all databases.

Naive Solution – Uniform Sampling: The simplest sampling framework is to uniformly allocate the S sample documents to each database, meaning that for each database an equal number of documents will be sampled, i.e. $\lfloor \frac{S}{n} \rfloor$, for $1 \leq i \leq n$. The uniform approach is fairly standard, and has been widely adopted, e.g., [85, 90, 160]. Such a uniform allocation is indifferent to the relative size of each database or the relative quality of the document samples. Thus, the number of sample documents collected from a relatively small database

like the technical jobs site Dice [http://www.dice.com] with its approximately 100,000 job listings is the same as the number of sample documents collected from the much larger and more diverse Monster site [http://www.monster.com] with its one million job listings in a variety of diverse fields. We argue that the simple uniform sampling approach misses the relative quality of each database sample with respect to the entire space of sampled databases (e.g., by sampling too few documents from Monster and too many from Dice) and tends to produce lower quality samples, severely affecting the overall quality and performance of the distributed query sampling framework. In this chapter we present an adaptive sampling framework that is quality-conscious and self-configuring and our experiments show that our approach is effective in producing higher quality samples and improving the performance and quality of distributed database selection services.

6.3 Adaptive Architecture for Distributed Query-Sampling

In contrast to the simple uniform sampling approach that makes no attempt to optimize the sampling quality for each text database based on the overall quality of all n databases, the main idea of our adaptive sampling framework is to dynamically determine the amount of sampling at each text database based on an analysis of the relative merits of each database sample *during* the sampling process. In practice, optimizing the overall quality of the database samples is difficult for a number of reasons. First, databases tend not to export detailed quality statistics for aiding text database sampling. Second, the dynamic sampling process typically does not have access to the entire database contents for evaluating the quality of the extracted database samples.

In this section we introduce a distributed and adaptive query-sampling framework and examine three quality-conscious sampling schemes to estimate critical database size and vocabulary parameters as a proxy for database quality to guide the sampling process. The challenging issue is the balance between the quality of each text database sample versus the quality of all database samples.

Table 10: Sampling Notation

S	total number of sample documents
S_{seed}	total number of sample documents for Seed Sampling
S_{dyn}	total number of sample documents for Dynamic Sampling
$s_{seed}(D_i)$	number of sample documents allocated for Seed Sampling of database D_i
$s_{dyn}(D_i, j)$	number of sample documents allocated for Dynamic Sampling of database D_i in iteration j
$s_{tot}(D_i)$	$s_{seed}(D_i) + \sum_{j=1}^{m} s_{dyn}(D_i, j)$
$s(D_i)$	ideal total number of sample documents that should be allocated to database D_i
$\hat{s}(D_i)$	estimated total number of sample documents that should be allocated to database D_i

6.3.1 Adaptive Sampling Steps: An Overview

The goal of the adaptive approach to the *distributed query-sampling problem* is to identify the optimal allocation of the S documents to the n databases. The challenging issue is the balance between the quality of each text database sample versus the quality of all database samples. The adaptive sampling approach to the distributed query sampling problem divides the query-based sampling process into three steps: the initial seed sampling, the dynamic quality conscious sampling allocation, and the dynamic sampling execution. The last two steps are typically implemented using an iterative approach, such that each sampling iteration dynamically allocates the amount of samples based on estimated database size and quality parameters derived during the previous sampling iteration.

1. Seed Sampling: In this step, we collect an initial seed sample from each database to bootstrap the iterative distributed query-sampling process. A portion of the total sample documents S is allocated for seed sampling, denoted by S_{seed}. One simple approach to selecting a seed sample is to use the uniform allocation of S to n databases under consideration, such that each database D_i $(i = 1, ..., n)$ is sampled using Query-Based Sampling until $s_{seed}(D_i) = \frac{S_{seed}}{n}$ documents are extracted.

2. Dynamic Sampling Allocation: This step utilizes the seed samples collected from each database to estimate various size and quality parameters of each participating database. The remaining number of sample documents is denoted by S_{dyn} where $S_{dyn} = S - S_{seed}$.

The adaptive sampling can be conducted in m iterations and in each iteration, S_{dyn}/m sample documents are dynamically allocated to n databases based on a Quality-Conscious Sampling scheme. When $m = 1$, there is no iteration. We denote the number of sample documents allocated to database D_i in iteration j as $s_{dyn}(D_i, j)$. We provide a detailed discussion of three concrete quality-conscious sampling schemes for guiding the dynamic sampling allocation in the subsequent section.

3. Dynamic Sampling Execution: In this step, the n databases are sampled according to the documents allocated by the Dynamic Sampling Allocation step using Query-Based Sampling. In the case of $m > 1$, steps 2 and 3 will be iterated m times. At the end of the Query Sampling Execution step, we will have sampled for each database D_i a total number of documents, denoted by $s_{tot}(D_i)$, such that $s_{tot}(D_i) = s_{seed}(D_i) + \sum_{j=1}^{m} s_{dyn}(D_i, j)$.

Table 10 summarizes the notation introduced in this section.[1]

6.3.2 Quality-Conscious Sampling Schemes

In this section, we focus the discussion on the problem of dynamic sampling allocation based on relative size and quality characteristics of the n participating databases. These quality statistics are estimated by analyzing the sampled documents from the previous sampling step. Concretely we conquer the dynamic sampling allocation problem in two stages. First, we describe three quality-conscious sampling schemes, and each scheme recommends for each D_i a total number of documents to sample, denoted by $\hat{s}(D_i)$, which is designed to be a close approximation of what would be recommend when the complete information about each database is available, denoted by $s(D_i)$. Then we will discuss how to find the dynamic sampling allocation $s_{dyn}(D_i)$ for each database based on $\hat{s}(D_i)$.

6.3.2.1 Scheme 1: Proportional Document Ratio [PD]

The first Quality-Conscious Sampling Scheme is based on the relative size of each database. Instead of simply collecting the same number of documents from each database as in the

[1]Note that we require that all sample sizes be integers since they correspond to the number of documents to be sampled from each database. In the rest of the chapter, we shall omit the rounding of sample sizes for presentation purposes.

naive uniform sampling approach, this scheme seeks to collect the same proportion of documents from each database, say 5% of the documents at D_1, 10% of the documents at D_2, and so on. The hypothesis is that document samples that better reflect the relative distribution of documents at each database will lead to higher-quality samples and hence, better application performance like database selection, categorization, and so on.

If we assume that the sampling framework has access to the total number of documents $|D_i|$ in each database D_i, $i = 1, ..., n$, then the proportion of documents to be sampled from each database is governed by the total documents available for sampling and the total size of the databases to be sampled: $ratio_{PD} = \frac{S}{\sum_{i=1}^{n} |D_i|}$.

Hence, the goal of the Proportional Document Ratio (PD) scheme is to sample the same fraction $ratio_{PD}$ from each database. So, the total number of documents to be extracted from database D_i is given by: $s_{PD}(D_i) = ratio_{PD} \cdot |D_i| = \frac{S}{\sum_{j=1}^{n} |D_j|} \cdot |D_i|$.

Of course, the actual number of documents in each database may not be known a priori. As a result, the PD scheme will typically rely on an estimate $|\tilde{D}_i|$ of the number of documents in database D_i, instead of the actual size $|D_i|$. Hence, we must estimate the fraction of documents to be extracted from each database with: $\widetilde{ratio}_{PD} = \frac{S}{\sum_{i=1}^{n} |\tilde{D}_i|}$. As a result, we may approximate $s_{PD}(D_i)$ with the estimate: $\hat{s}_{PD}(D_i) = \widetilde{ratio}_{PD} \cdot |\tilde{D}_i| = \frac{S}{\sum_{j=1}^{n} |\tilde{D}_j|} \cdot |\tilde{D}_i|$.

Another important quality measure is the database size estimates which can be calculated after the Seed Sampling step by analyzing the extracted document samples. There have been two previous efforts to estimate the number of documents in a text database: (1) the capture-recapture algorithm in [110]; and (2) the sample-resample algorithm in [161]. Both approaches rely on analyzing a document sample, but the sample-resample approach has been shown to be more accurate and less expensive in terms of queries and the number of documents necessary to sample.

The sample-resample technique assumes that the database responds to queries by indicating the total number of documents in which the query occurs (although only a fraction of this total will be available for download by the client). The "sample" phase collects a document sample (say 300 documents), then the "resample" phase issues a handful of additional queries and collects the $c(t, D)$ statistics for these queries. The driving assumption of this

technique is that the fraction of sampled documents that contain a term is the same as the fraction of all documents in the database that contain the same term, i.e. $\frac{c(t,D_s)}{|D_s|} = \frac{c(t,D)}{|\tilde{D}|}$, where the database provides $c(t, D)$ or a reasonable approximation. Hence, $|\tilde{D}| = \frac{|D_s| \cdot c(t,D)}{c(t,D_s)}$. This database size estimate may be further refined by considering a set of p probe terms $(t_1, ..., t_p)$ and taking the average estimate: $|\tilde{D}| = \frac{1}{p} \sum_{i=1}^{p} \frac{|D_s| \cdot c(t_i,D)}{c(t_i,D_s)}$. In practice, we collect the $c(t, D)$ statistics for all queries issued to the database.

The original sample-resample algorithm suggests sending queries that are randomly selected from the sampled documents – in our experiments we have seen that this technique may grossly underestimate $|D|$. Since term occurrences follow a Zipfian distribution, many of the terms in the sampled documents will be infrequent, and hence, a simple random selection will tend to oversample rare terms, which can lead to poor estimates of the number of documents. For example, suppose we have a sample of 300 documents for a database consisting of 100,000 documents. A choice of a probe term that occurs in only 1 document in the database (and 1 document in our sample) will result in a size estimate for the total database of 300, an underestimation error of over 99%. In contrast, we use a weighted random selection of resample probes based on the frequency of the terms in the sampled documents.

6.3.2.2 Scheme 2: Proportional Vocabulary Ratio [PV]

The second quality-conscious sampling scheme is similar in spirit to the Proportional Document Ratio scheme. Instead of collecting the same proportion of documents from each database, the Proportional Vocabulary Ratio (PV) scheme seeks to sample the same *vocabulary* proportion from each database, say 10% of the vocabulary terms at D_1, 10% of the vocabulary terms at D_2, and so on. Unlike the first scheme, the PV scheme is intuitively more closely linked to the quality of the database samples, since it emphasizes the presence of *unique* terms, and not just document quantity.

Assume for a moment that we have complete access to the documents in all databases. Given this information, we know the total vocabulary size $|V_i|$ for each database D_i. The PV sampling scheme must then determine the fraction of documents necessary to sample

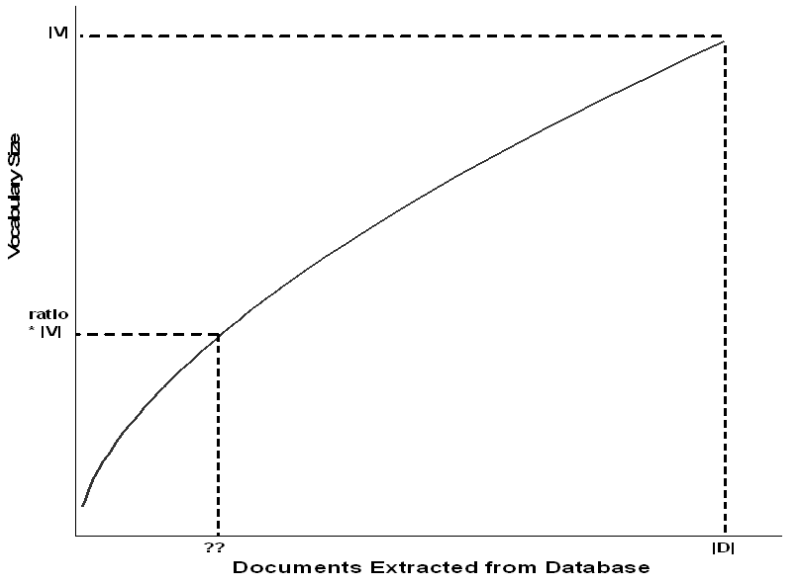

Figure 65: How Many Documents Should be Sampled to Yield $ratio_{PV} \cdot |V|$ Vocabulary Terms?

that will yield some fraction $ratio_{PV}$ of the total vocabulary for each database. That is, how many documents should be sampled to extract $ratio_{PV} \cdot |V_i|$ vocabulary terms for each database D_i?

According to Heaps Law [13], a text of n words will have a vocabulary size of Kn^{β}, where K and β are free parameters and typically $0.4 \leq \beta \leq 0.6$. What this means is that there are diminishing returns of vocabulary terms as the text size grows larger. In the context of distributed text databases, documents sampled earlier in the sampling process will tend to contribute more new vocabulary terms than will documents that are sampled later in the process. As a result, the PV scheme must rely on a more sophisticated approach than the linear one of the PD scheme.

Suppose we randomly examine documents in a database and then plot the number of documents examined versus the vocabulary size, yielding a Heaps Law curve. If we were to repeat this process, we would derive a Heaps Law curve similar to the one illustrated in Figure 65. To identify the number of sample documents $s_{PV}(D_i)$ necessary to extract $ratio_{PV} \cdot |V_i|$ vocabulary terms, we need only consult the Heaps Law curve to identify $s_{PV}(D_i)$. Due to the inherently random nature of the sampling process and since documents

in a text database can be of different lengths and the number of vocabulary terms may vary from document to document, this technique provides an average-case estimate of the number of documents necessary to achieve a particular fraction of all vocabulary terms.

In practice, the sampling framework must rely only on the seed sample to estimate the vocabulary fraction and the corresponding number of documents necessary to achieve the fraction $ratio_{PV} \cdot |V_i|$. To estimate the number of documents necessary requires that we carefully inspect the sampled documents in the seed sample. We may begin by approximating the total size of the vocabulary $|V|$ of D based on a sample of documents D_s by relying on a version of Heaps Law adapted to the distributed text database domain:

$$|V| = K\,(f(D))^{\beta}$$

where $f(D) = \sum_{t \in V} f(t, D)$ refers to the total frequency of all terms in D. The keys for the vocabulary estimation are now reduced to identifying the appropriate values for K, β, and $f(D)$ based on the sample documents only. Hence, we will find K_s, β_s, and $f(\tilde{D})$ respectively by analyzing the sample documents only.

Estimating $f(D)$: First, we will discuss how to approximate $f(D)$ with $f(\tilde{D})$. If we let $|d|_{avg}$ denote the average number of terms per document for the entire database, then we may write $f(D)$ as a product of $|d|_{avg}$ and the total number of documents in the database: $f(D) = |d|_{avg} \cdot |D|$. Since we showed how to approximate $|D|$ with $|\tilde{D}|$ in the previous section, we need only approximate $|d|_{avg}$ in order to find $f(\tilde{D})$. We can estimate the average number of terms per document $|d|_{avg}$ for the entire database by examining the seed sample: $|\tilde{d}|_{avg} = \frac{f(D_s)}{|D_s|}$, where $f(D_s)$ refers to the total frequency of all terms in the document sample D_s: $f(D_s) = \sum_{t \in V_s} f(t, D_s)$. Hence, we have $f(\tilde{D}) = |\tilde{d}|_{avg} \cdot |\tilde{D}|$.

Estimating K and β: Next, we will show how to approximate the Heaps Law parameters K and β with K_s and β_s, respectively. To do so, we consider increasingly larger sub-samples of the total set of sampled documents. In particular, we randomly select a document (without replacement) from the set of sampled documents D_s and add it to the sub-sample D_{ss}. For each sub-sample D_{ss}, we plot the total term frequency (i.e. text size) $f(D_{ss})$ versus the actual vocabulary size $|V_{ss}|$ of the sub-sample. This process repeats until all documents

in the sample have been selected. As a result, we will have $|D_s|$ points consisting of the text size and the corresponding vocabulary size. We may then estimate the values of K_s and β_s by using a curve fitting tool (like the FindFit function available in Mathematica) to fit the best function that conforms to Heaps Law to these $|D_s|$ points.

With these estimated parameters, the total vocabulary size $|V|$ of the database may be approximated with the estimate $|\tilde{V}|$ by calculating:

$$|\tilde{V}| = K_s(|f(\tilde{D})|)^{\beta_s}$$

Hence, by analyzing only the documents in the Seed Sample step, we may estimate the total vocabulary size of all n text databases.

Recall that the Proportional Vocabulary Ratio scheme seeks to equalize the quality of the database samples for each database by extracting the same fraction of the estimated vocabulary from each database. Hence, we need to find the sample size $\hat{s}_{PV}(D)$ that should yield $ratio_{PV} \cdot |\tilde{V}|$ vocabulary terms at a database. Relying on the Heaps Law parameter estimates (K_s and β_s) and the average size of each document ($|\tilde{d}|_{avg}$) for each database, we have n equations of the form:

$$ratio_{PV} \cdot |\tilde{V}| = K_s(|\tilde{d}|_{avg} \cdot \hat{s}_{PV}(D))^{\beta_s}$$

Solving for $\hat{s}_{PV}(D)$ yields:

$$\hat{s}_{PV}(D) = e^{\frac{\ln ratio_{PV}|\tilde{V}| - \ln K_s}{\beta_s} - \ln |\tilde{d}|_{avg}}$$

Given the n equations for $\hat{s}_{PV}(D_i)$ (one for each database), we need to determine the appropriate choice of $ratio_{PV}$ such that the total documents to be sampled is $S = \sum_1^n |\hat{s}_{PV}(D_i)|$, where $0 \leq ratio_{PV} \leq 1$. Since each equation is monotonically increasing with respect to the choice of $ratio_{PV}$, we may solve the non-linear optimization problem using the simple search approach in Figure 66.

As a result, we may find for each database the estimated number of documents necessary to extract some proportion ($ratio_{PV}$) of all vocabulary terms at the database.

SearchRatioPV
Initialize $MIN = 0$ and $MAX = 1$
repeat
Randomly choose $ratio_{PV}$ from (MIN, MAX)
Evaluate $S' = \sum_1^n |\hat{s}_{PV}(D_i)|$
if $S' < S$, let $MIN \leftarrow ratio_{PV}$
else, let $MAX \leftarrow ratio_{PV}$
until $|S - S'| < \epsilon$
return $ratio_{PV}$

Figure 66: Simple Search Strategy to Find RatioPV

6.3.2.3 Scheme 3: Vocabulary Growth [VG]

The final sampling scheme is based on the vocabulary growth rate at each database. Rather than seek to extract the same proportion of vocabulary terms from each database as in the Proportional Vocabulary Ratio scheme, the goal of the Vocabulary Growth (VG) scheme is to extract the *most* vocabulary terms from across the space of distributed text databases.

This scheme relies on the Heaps Law parameter estimates κ_s and β_s for each database, as we presented in the previous section. By analyzing how fast each database "produces" new vocabulary terms as more documents are sampled, our goal is to extract the "cheapest" vocabulary terms first. Some databases may have a very slow growing vocabulary, meaning that many more documents must be sampled to equal the same number of vocabulary terms as a database with a fast growing vocabulary. Due to the nature of the Heaps Law curve, at some point the additional vocabulary terms for each sampled document may slow to a point that additional sampling is not worthwhile in the context of many distributed text databases.

To guide the VG sampling scheme, we must first understand the vocabulary growth rate at each database. To estimate the growth rate of the vocabulary size, we may consider the incremental vocabulary terms that each additional sampled document may be expected to yield. If we let x denote the number of documents sampled from a database through query-based sampling, then we can write the estimated vocabulary size for the x documents as $|V(x)|$:

$$|V(x)| = K_s(|\tilde{d}|_{avg} \cdot x)^{\beta_s}$$

To determine the incremental vocabulary terms – denoted by $\Delta(|V(x)|)$ – available by sampling x documents versus sampling $x-1$ documents, we take the difference between the expected vocabulary size for a sample of size x documents and for a sample of size $x-1$:

$$\Delta(|V(x)|) = |V(x)| - |V(x-1)|$$
$$\Delta(|V(x)|) = K_s(|\tilde{d}|_{avg} \cdot x)^{\beta_s} - K_s(|\tilde{d}|_{avg} \cdot (x-1))^{\beta_s}$$

Hence, we may determine the expected incremental vocabulary terms of each successive sampled document. For each of the databases, we may calculate the incremental vocabulary terms $\Delta(|V(x)|)$ for all documents available at the database. If we consider the "price" of each additional vocabulary term extracted from a database as the number of documents per new vocabulary term, then we may choose to sample from each database based on the "cheapest" database for extracting new vocabulary terms. Equivalently, we choose from which database to sample based on how many additional vocabulary terms it is expected to yield per document sampled. With a total number of documents to sample of S, we select the top-S documents from across all databases as scored by $\Delta(|V(x)|)$. We then allocate to each database $\hat{s}_{VG}(D_i)$ documents based on the total number of D_i's documents in the top-S.

Example: To illustrate the VG sampling scheme, consider a simple example scenario with three databases – A, B, and C, shown in Figure 11. Suppose for each we have estimated the appropriate κ and β parameters describing the vocabulary growth curve for each. We then show the incremental vocabulary terms expected from sampling one to ten documents. Given a total budget for sampling 10 documents, we show in bold the ten documents with the highest incremental vocabulary terms. As a result, the VG sampling scheme would extract 5 documents from A, 2 from B, and 3 from C.

Table 11: Example Vocabulary Growth Scenario

Doc	$A(\kappa = 100;$ $\beta = 0.6)$	$B(\kappa = 120;$ $\beta = 0.4)$	$C(\kappa = 80;$ $\beta = 0.6)$
1	**100.0**	**120.0**	**80.0**
2	**51.6**	**38.3**	**41.3**
3	**41.7**	27.9	**33.4**
4	**36.4**	22.7	29.1
5	**32.9**	19.5	26.3
6	30.4	17.3	24.3
7	28.4	15.6	22.7
8	26.8	14.3	21.4
9	25.5	13.3	20.4
10	24.4	12.4	19.5

Unlike the PD and PV sampling schemes, the Vocabulary Growth scheme is concerned not with the relative size of each database, but in the "efficiency" of each database with respect to how fast each produces new terms as documents as sampled. As a result, the VG scheme does not rely on estimating either the total number of documents or the total vocabulary size for each database.

6.3.3 Dynamic Sampling Allocation

Recall that given a collection of n databases and the maximum S documents to be sampled, our distributed adaptive the query-sampling framework divides the total number S of documents to be sampled into an amount for Seed Sampling S_{seed} and the amount for Dynamic Sampling Allocation & Execution, namely $S = S_{seed} + S_{dyn}$. As a result, each database D_i receives a total allocation of $s_{tot}(D_i) = s_{seed}(D_i) + s_{dyn}(D_i)$ for $i = 1, \ldots, n$. We have described three quality-conscious sampling schemes, which can be used to recommend an approximation of the ideal number of documents that should be allocated to each database when complete information about D_i is available, denoted by $\hat{s}(D_i)$. Now we need to determine the number of documents to be sampled from each of the n databases for dynamic sampling, denoted by $s_{dyn}(D_i)$, such that the total documents sampled adaptively from each remaining database, denoted as $s_{tot}(D_i)$, closely matches the number of documents prescribed by the sampling scheme, that is: $\hat{s}(D_i) \approx s_{tot}(D_i)$. There are two cases to consider:

Case 1. If for all databases, the seed sampling step has extracted fewer documents than the scheme recommends in total (i.e., $\hat{s}(D_i) > s_{seed}(D_i)$), then we simply let the Dynamic Execution step sample the remaining documents $s_{dyn}(D_i) = \hat{s}(D_i) - s_{seed}(D_i)$.

Case 2. However, it may be the case that a database has already been oversampled in the Seed Sampling step with respect to the sampling recommendation by the quality-conscious sampling scheme, i.e., $\hat{s}(D_i) < s_{seed}(D_i)$. This oversampling requires two corrections. First, any database that has been sampled sufficiently in the Seed Sampling step is dropped from the Dynamic Sampling Execution step. Second, we must re-scale the documents allocated to the remaining databases. A number of scaling factors can be considered, assuming that $\mathcal{A}$ denotes the set of databases under consideration by the Dynamic Sampling Allocation step. In this chapter, we use a simple scaling:

$$s_{dyn}(D_i) \leftarrow S_{dyn} \frac{\hat{s}(D_i) - s_{seed}}{\sum_{D_k \in \mathcal{A}} (\hat{s}(D_k) - s_{seed})}$$

Example: To illustrate this scaling, consider an example scenario, shown in Figure 12. There are five databases, ranging in size from 20,000 to 500,000 documents. Suppose the framework may sample a total of 10,000 documents. With perfect information the PD sampling scheme would result in an ideal sampling allocation as shown in the column labelled $s(D_i)$. The goal of the PD sampling scheme is to closely emulate this ideal allocation. In the Seed Sampling step, half of the sampling documents are uniformly distributed for each database, meaning 1,000 documents are sampled from each [column $s_{seed}(D_i)$], leaving 5,000 documents for the Dynamic Sampling Allocation and Dynamic Sampling Execution steps. Suppose the sampling scheme has estimated the appropriate database parameters and has derived the estimate $\hat{s}(D_i)$ for each database, which in this case correlate closely with the ideal sampling allocation. In this case, two of the databases (D and E) have been oversampled already by the Seed Sampling step ($1,000 > 600$, and $1,0000 > 300$). Hence, we need consider only database A, B, and C for the Dynamic Sampling Execution step. Based on the scaling factor described above, we can find $s_{dyn}(D_i)$ for each database, resulting in the total sample sizes shown in column $s_{tot}(D_i)$.

Table 12: Example Sampling Scenario: PD Sampling Scheme

D_i	$\lvert D_i \rvert$	$\hat{s}(D_i)$	$s_{seed}(D_i)$	$s_{dyn}(D_i)$	$s_{tot}(D_i)$
A	500,000	5,100	1,000	3,360	4,360
B	300,000	2,900	1,000	1,560	2,560
C	130,000	1,100	1,000	80	1,080
D	50,000	600	1,000	0	1,000
E	20,000	300	1,000	0	1,000
Total	1,000,000	10,000	5,000	5,000	10,000

6.3.4 Adaptive Sampling: Multi-Round Iterations

So far we have described in detail the basic distributed query-sampling framework, which considers a Seed Sampling step, followed by a single round of Dynamic Sampling Allocation and Dynamic Sampling Execution. A single round dynamic sampling means that each quality-conscious sampling scheme makes a recommendation of document sampling allocation only once, immediately after the Seed Sampling step. One obvious drawback of single round adaptive sampling is the situation where the initial seed samples do not reflect the true underlying database contents. Therefore the Dynamic Sampling Allocation step may misallocate the remaining documents relative to the actual database contents.

An effective approach to address this problem is to extend the basic algorithm by supporting multiple rounds of Dynamic Sampling Allocation and Dynamic Sampling Execution. The multi-round iteration bases approach allocates the sample documents available for dynamic sampling S_{dyn} over multiple (r) rounds of dynamic sampling, where each round is allocated S_{dyn}/r total documents. Since each sampling scheme depends on database size and quality estimates based on an analysis of the extracted document samples, we would expect that increasing the number of rounds would result in more refined dynamic sampling, and reducing the impact of errors generated in the seed sampling step on the quality of adaptive sampling phase.

6.4 Experiments

In this section, we present three sets of experiments designed to test the distributed query-sampling framework. The experiments rely on data drawn from two standard TREC information retrieval datasets summarized in Table 13.

Table 13: Overall TREC Dataset Summary Information

Name	Size (GB)	Documents	Total Terms
TREC123	3.2	1,078,166	258,212,077
TREC4	2.0	567,529	155,575,164

TREC123:: This set consists of 100 databases created from TREC CDs 1, 2, and 3. The databases are organized by source and publication date as described in [141].

TREC4: This set consists of 100 databases drawn from TREC 4 data. The databases correspond to documents that have been clustered by a k-means clustering algorithm using a KL-divergence based distance measure as described in [184]. By construction, each database is topically similar.

Detailed information about the 100 component databases can be found in Table 14.

Table 14: Per Database Summary Information

		Documents			Vocab ('000s)	
Name	Min	Avg	Max	Min	Avg	Max
TREC123	752	10,782	39,731	27.4	63.8	92.4
TREC4	301	5,675	87,727	9.8	35.4	200.0

In addition, we created six large databases to further test the distributed query-sampling framework. The large databases were created from TREC123 data and are listed in Table 15. The large database AP is composed of all 24 databases of Associated Press articles in the TREC123 dataset. Similarly, WSJ is composed of the 16 Wall Street Journal databases; FR the 13 Federal Register databases; and DOE the 6 Department of Energy databases. The two other databases – Rand1 and Rand2 – are each combinations of 20 non-overlapping randomly selected databases from TREC123. Based on these large databases, we created three additional datasets.

Table 15: Large TREC Datasets

Name	Documents	Vocab Size	Total Terms
AP	242,918	347,762	61,381,800
WSJ	173,252	314,791	43,542,976
FR	45,820	503,774	34,588,476
DOE	226,087	206,653	16,874,516
Rand1	232,031	544,558	50,129,684
Rand2	228,701	553,007	50,152,660

TREC123-A: This dataset consists of the large AP and WSJ databases, plus the 60 other trec123 databases (excluding the AP and WSJ databases), for a total of 62 databases. The AP and WSJ databases are much larger than the other databases and contain a disproportionate share of relevant documents for the tested query mix for the database selection application scenario.

TREC123-B: This dataset consists of the large FR and DOE databases, plus the 81 other trec123 databases (excluding the FR and DOE databases), for a total of 83 databases. The FR and DOE databases are much larger than the other databases, but contain very few relevant documents for the tested query mix.

TREC123-C: This dataset consists of the large Rand1 and Rand2 datasets, plus the 60 other trec123 databases, for a total of 62 databases. By construction, the Rand1 and Rand2 datasets contain approximately the same proportion of relevant docs as all the other databases.

All database sampling and selection code was written in Java. The curve fitting necessary for parameter estimation was performed with Mathematica via the Java interface J/Link. Each dataset was indexed and searched using the open source Lucene search engine. The search engine indexes and database samples do not include a list of standard stopwords; the terms have been stemmed using the standard Porter's Stemmer [140]. In all cases the Query-Based Sampling component relied on a simple random prober that drew probe terms initially from the standard UNIX dictionary and subsequently from the sampled documents; a maximum of four documents were retrieved for each query.

6.4.1 Estimation Error

In the first set of experiments, we study how effectively the sampling framework may estimate the critical database size parameters necessary to drive the three sampling schemes. Depending on the sampling scheme, the seed sample is used to estimate either (1) the number of documents at each database [for the PD scheme]; or (2) the total vocabulary size of each database and vocabulary-related parameters κ and β [for the PV and VG schemes]. In this set of experiments, we measure the relative error for estimating the total size $|D|$ of

each database and the vocabulary size $|V|$.

There are number of open issues with respect to this estimation that we seek to answer with this set of experiments: (1) Does the estimation error fall as the sample size increases?; (2) Does the error fall quickly? Or does it require samples of such considerable size that the estimation is infeasible in practice?; and (3) Does the estimation technique work well across databases of different sizes and characteristics?

We measure the relative error for the database size as $E_D = \frac{|\tilde{D}|-|D|}{|D|}$, where $|D|$ denotes the actual database size in documents and $|\tilde{D}|$ denotes the estimated database size based on the sampled documents. Similarly, we measure the relative vocabulary size error as $E_V = \frac{|\tilde{V}|-|V|}{|V|}$, where $|V|$ denotes the vocabulary size of the actual database and $|\tilde{V}|$ denotes the estimated vocabulary size based on the sampled documents and the parameter estimation. To assess the quality of the parameter estimation, we measure the error rate across the 100 TREC4 databases, the 100 TREC123 database, and the six large databases described in Table 15. For each database, we extracted from 50 to 500 documents and calculated the error rate for the size estimates. We repeated this process five times and report the average.

We begin by reporting the database size error E_D. Across the TREC4 and TREC123 databases, as the sample size increases from 50 documents to 500, the estimation error quickly becomes reasonable. For TREC4, the relative error ranges from 13% to 18%. Similarly, for TREC123, the relative error ranges from 14% to 18%. For the large databases, the database size error ranges from 20% to 30%, on average. These results validate the results from the original sample-resample chapter [161] and provide strong evidence that the database size may be estimated for large database by examining only a small fraction of the documents.

The vocabulary size estimation is significantly more difficult since it relies on estimating multiple parameters, including the κ and β parameters for the Heaps Law curve, as well as the average size of each document in the database. In Figures 67 and 68 we report the average vocabulary estimation error for the 100 TREC4 databases, the 100 TREC123 databases, and the 6 large databases. Since the vocabulary size estimation relies on knowing the number of documents in a database, we report the vocabulary error in the ideal case

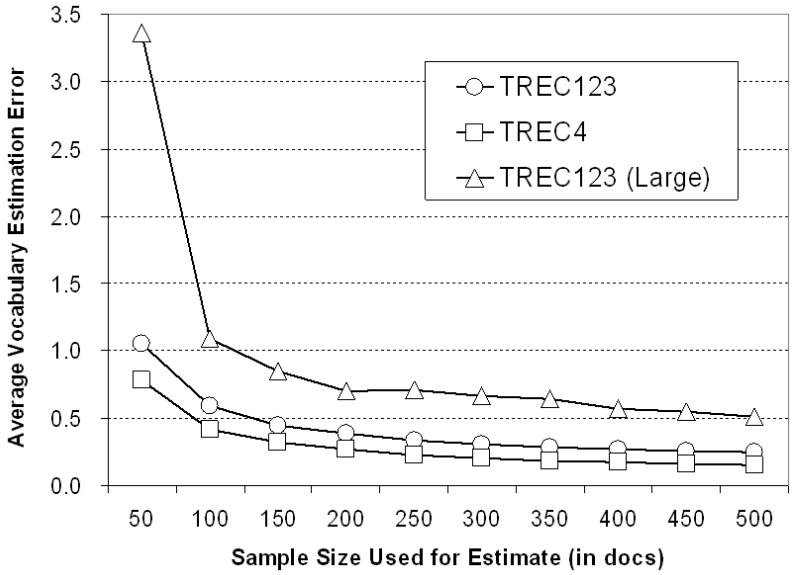

Figure 67: Vocabulary Estimation Error - Database Size Known

when the database size is known and in the more realistic case when the database size must be estimated. The results are encouraging. In all cases, the error falls quickly, requiring sample sizes of fewer than 200 documents for reasonably accurate quality estimates. For example, both the TREC4 and TREC123 estimates are within 50% of the actual after examining only 150 documents. For the large databases, the error is within a factor of 2 after only 100 documents.

To further understand the vocabulary estimation error, we report in Figures 69 and 70 the cumulative probability distribution of the average vocabulary estimation error for TREC4 and TREC123. Each line represents the error distribution based on a particular sample size. So the line "150" indicates the error distribution when the vocabulary estimate is made after sampling 150 documents. More than half of the database have an estimation error of less than 0.30, and 90% of the databases have an error less than 0.50 after sampling only 150 documents. As the sample size used for estimation increases, the quality of the vocabulary estimates gets better. Similar results hold for the large databases.

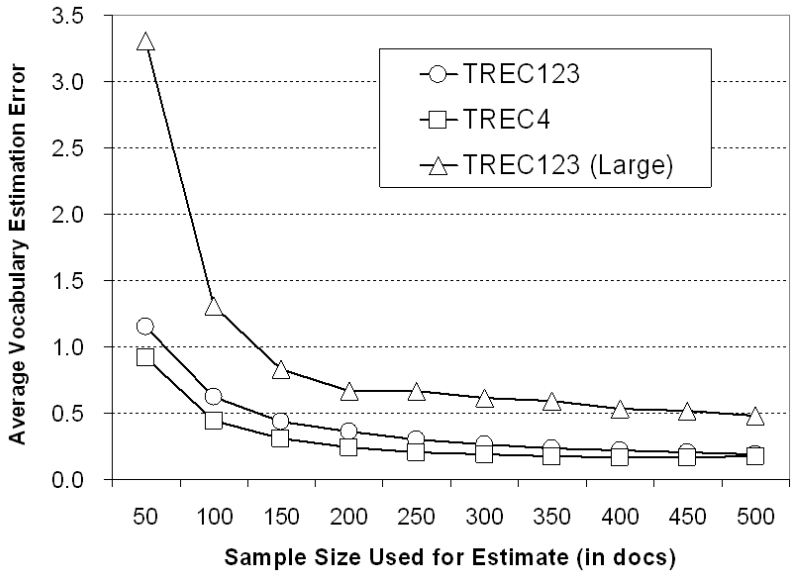

Figure 68: Vocabulary Estimation Error - Database Size Estimated

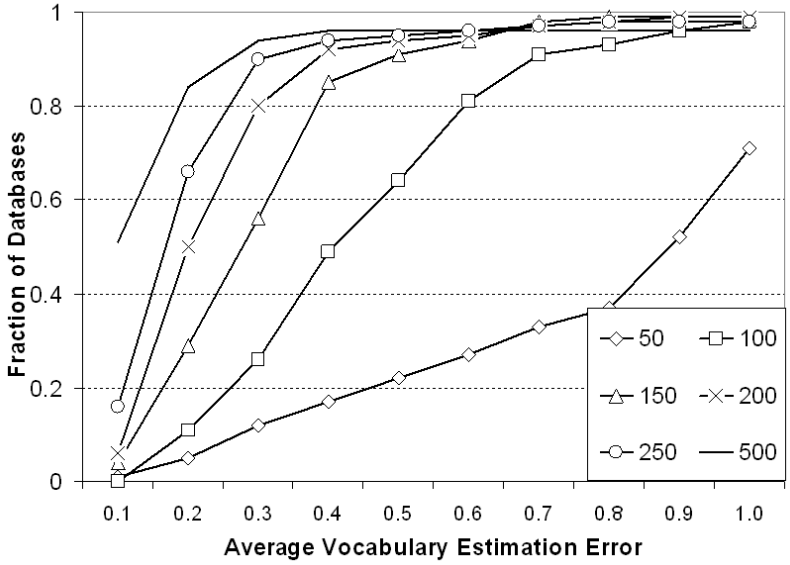

Figure 69: Cumulative Error Distribution [Estimated Docs], TREC4

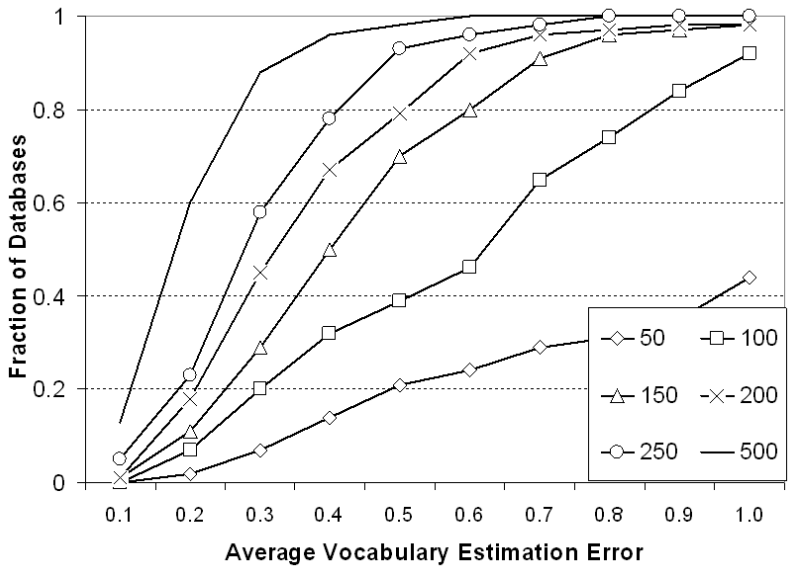

Figure 70: Cumulative Error Distribution [Estimated Docs], TREC123

6.4.2 Database Sample Quality

As we have seen, the distributed query-sampling framework may estimate the database size and vocabulary parameters using fairly small seed sample sets. In this next set of experiments, we study the impact on the overall sample quality of the three quality-conscious sampling schemes – Proportional Document Ratio (PD), Proportional Vocabulary Ratio (PV), and Vocabulary Growth (VG) – versus the uniform sampling approach. Our goal in this experiment is to understand whether the three schemes are capable of extracting higher-quality database samples than the uniform sampling approach.

Since previous research efforts have claimed that 300 documents are reasonable for extracting high-quality database samples from databases ranging in size from thousands of documents to millions (e.g., [37, 36]), for each of the datasets we assumed a total document budget of $S = 300 \cdot n$, where n is the number of databases in the dataset. So, the total budget for TREC123 (100 databases) is 30,000 documents, for TREC4 (100 databases) 30,000, for TREC123-A (62 databases) 18,600, for TREC123-B (83 databases) 24,900, and for TREC123-C (62 databases) 18,600 total documents.

For each of the datasets, we collected baseline document samples using the Uniform sampling approach (U), meaning that 300 documents were sampled from each database.

For the three quality-conscious sampling schemes – Proportional Document Ratio (PD), Proportional Vocabulary Ratio (PV), and Vocabulary Growth (VG) – we allocated half of the total budget for the Seed Sampling step (i.e., 150 documents per database). We then allocated the remaining half of the total budget based on the specific sampling scheme. In this first set of sample quality experiments we consider a single round of dynamic sampling. Based on the seed samples alone, the schemes calculated the appropriate allocation of documents for the dynamic sampling.

6.4.2.1 Sample Quality Metrics

To assess the quality of the database samples produced by each sampling scheme, we consider a suite of six distinct quality metrics. Each metric compares a database sample D_s to the database D from which it is drawn.

Common Terms: This first metric measures the degree of term overlap between a database sample and the actual database contents:

$$ct(D_s, D) = \frac{|V \cap V_s|}{|V|}$$

where again V denotes the vocabulary of database D and V_s denotes the vocabulary of the database sample D_s.

Weighted Common Terms: Since the common terms metric disregards the relative frequency of each term, we next consider a weighted version:

$$wct(D_s, D) = \frac{\sum_{t \in V \wedge V_s} f(t, D)}{\sum_{t \in V_s} f(t, D)}$$

which is also known as the ctf_{ratio}[36].

Term Rankings: To assess the quality of the relative frequency of terms in the database sample, we rely on the Spearman rank correlation coefficient as defined in [36]. The Spearman coefficient measures the level of agreement between two rankings. In our case, we compare the rankings induced by ordering the terms in the actual database by $c(t, D)$ with the rankings induced by ordering the terms in the database sample by $c(t, D_s)$. The Spearman coefficient measures only the quality of the relative ranking assignment, not the values

assigned to each term. If both the database and the sample rank every term in the same position, then the Spearman coefficient is 1. Uncorrelated rankings result in a Spearman coefficient of 0; reverse rankings (i.e., the top-ranked term in the database is the lowest-ranked term in the database sample) result in a Spearman coefficient of -1.

Vector Similarity: The fourth quality metric is based on the vector space cosine similarity metric, which is often used to compare documents to documents, and queries to documents. Here, we apply the cosine to measure the vector space similarity of the database sample and the database:

$$cos(D_s, D) = \frac{\sum_{t \in V \cup V_s} c(t, D)c(t, D_s)}{\sqrt{\sum_{t \in V} c(t, D)^2}\sqrt{\sum_{t \in V_s} c(t, D_s)^2}}$$

The cosine ranges from 0 to 1, with higher scores indicating a higher degree of similarity. In contrast, the cosine between orthogonal vectors is 0, indicating that they are completely dissimilar. The cosine measures the angle between two vectors, regardless of the length of each vector.

Distributional Similarity: To measure the distributional similarity of the database sample and the actual database, we rely on the Jensen-Shannon divergence (or JS-divergence) [109]. It is based on the relative entropy measure (or KL-divergence), which measures the difference between two probability distributions p and q over an event space X: $KL(p, q) = \sum_{x \in X} p(x) \cdot \log(p(x)/q(x))$. Adopting a probabilistic interpretation, we can consider a text database D as a source randomly emitting a term t according to the overall prevalence of t in D. Hence, we can define $p(t|D) = \frac{f(t,D)}{f(D)}$ – where $f(D)$ refers to the total frequency of all terms in D: $f(D) = \sum_{t \in V} f(t, D)$. Hence, $kl(D, D_s) = kl(p, q) = \sum_{t \in V} p(t|D) \cdot \log \frac{p(t|D)}{q(t|D)}$. Intuitively, the KL-divergence indicates the inefficiency (in terms of wasted bits) of using the q distribution to encode the p distribution. Unfortunately, when comparing a database sample that lacks a single term from the actual database (which is almost always the case), the KL-divergence will be unbounded and, hence, will provide little power for evaluating database samples. In contrast, the Jensen-Shannon divergence avoids these problems. The JS-divergence is defined as:

$$js(D_s, D) = js(p, q) = \alpha_1 kl(p, \alpha_1 p + \alpha_2 q) + \alpha_2 kl(q, \alpha_1 p + \alpha_2 q)$$

where $\alpha_1, \alpha_2 > 0$ and $\alpha_1 + \alpha_2 = 1$. In our case, we consider $\alpha_1 = \alpha_2 = 0.5$. The lower the JS-divergence, the more similar are the two distributions. Unlike the KL-divergence, the JS-divergence is symmetric and does satisfy the triangle inequality, meaning that it is a metric. Additionally, the JS-divergence is bounded, so meaningful comparisons may be made between summaries that differ in the presence of certain terms (as is almost always the case).

The "Information" of a Text Database: The final quality metric measures the *information* of the sampled documents using a variation of the classic entropy measure. The entropy (or self-information) of a random variable X is defined as $H(X) = -\sum_{x \in X} p(x) \log p(x)$, where $H(X) \geq 0$ with higher values indicating more information (or randomness). Adopting a probabilistic interpretation, we can define the entropy of a database sample as:

$$H(D_s) = -\sum_{t \in V_s} p(t|D_s) \log_2 p(t|D_s)$$

where $p(t|D_s) = \frac{f(t,D_s)}{f(D_s)}$. The maximum possible entropy for a sample increases as the number of terms in the sample increases. So a document sample that explores a richer space of the database (with respect to the number of terms) will have the opportunity to yield a higher entropy, $0 \leq H(D_s) \leq log_2(|V_s|)$. Unlike the other quality metrics, the entropy does not require knowledge of the actual database contents to be evaluated. As we will see, the entropy is strongly correlated with many of the evaluation metrics that do require knowledge of the actual database contents, and so we anticipate further study of it.

For each of the six quality metrics, we measure the overall quality of the collected database samples for a dataset by calculating the average quality metric weighted by the actual size of the component databases for the dataset: $\sum_{i=1}^{n} \frac{Q(D_s,D)}{|D_i|}$.

6.4.2.2 Sample Quality Results

In Figures 71 to 76, we compare the Uniform (U) sampling approach versus the three quality-conscious sampling schemes of the sampling framework – Proportional Document

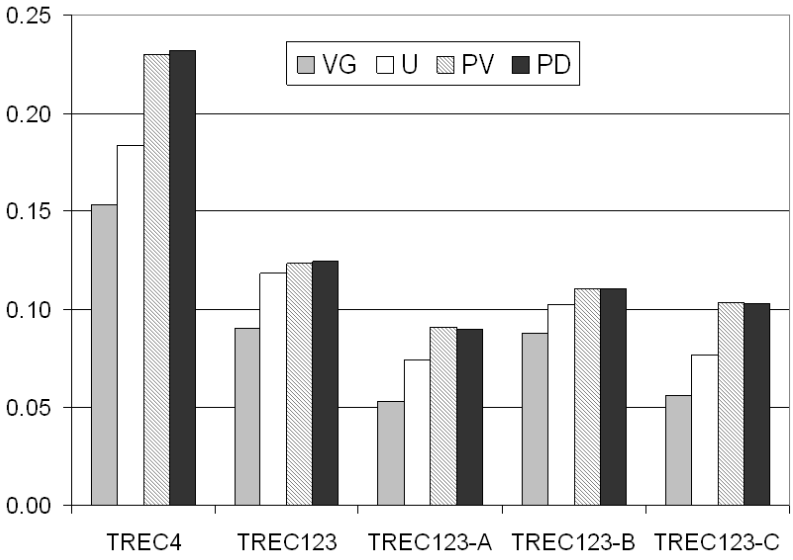

Figure 71: Common Terms (Summary Quality)

Ratio (PD), Proportional Vocabulary Ratio (PV), and Vocabulary Growth (VG). To minimize the randomness inherent in any sampling-based approach, we report the average results based on repeating the document sampling a total of five times for all tested schemes.

We note several interesting results. First, even under the strong constraint that the sampling schemes must rely solely on the seed samples for guiding the rest of the sampling process, we see that the PV and PD schemes outperform the uniform sampling approach U over *all five datasets* and *all six quality metrics*, indicating that the distributed query-sampling framework can yield higher quality samples based on multiple measures of sample quality. These results are encouraging since they indicate that there are significant opportunities for improving sample quality.

Second, the VG scheme significantly underperforms the U approach in all cases. On inspection, we discovered that the VG scheme resulted in an overall collection vocabulary of from 1.5 to 3 times as many vocabulary terms versus the other approaches across all settings. In Figure 77, we chart the total collection vocabulary size for each dataset. What is interesting here is that simply extracting more vocabulary terms from a collection of databases is not necessarily correlated with overall high-quality database samples. As we would expect, the VG scheme was very effective at extracting the most vocabulary terms of

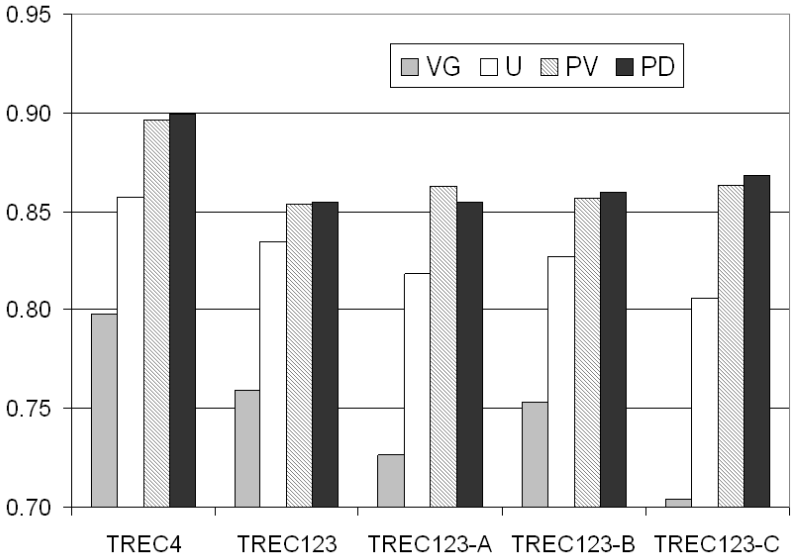

Figure 72: Weighted Common Terms (Summary Quality)

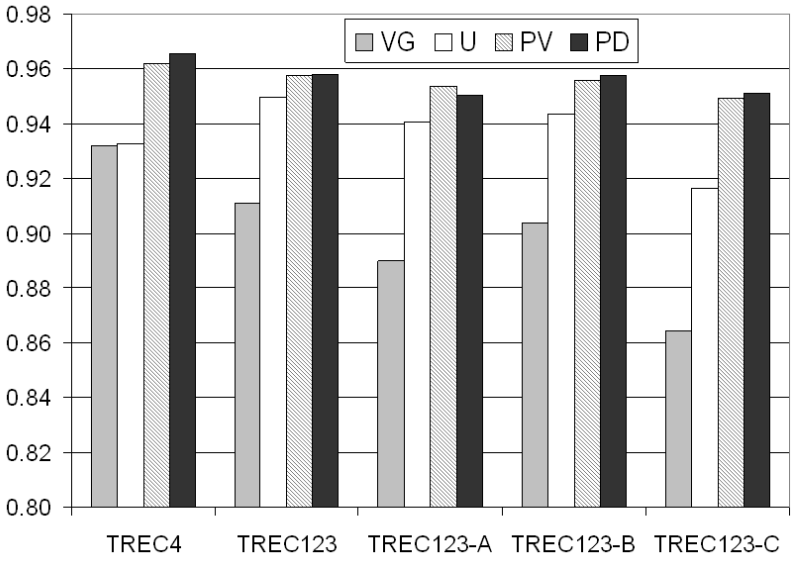

Figure 73: Cosine (Summary Quality)

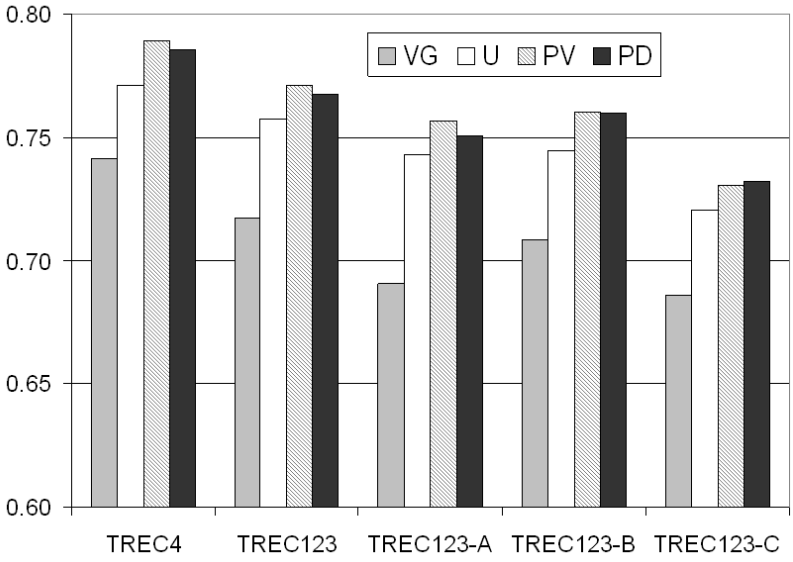

Figure 74: Spearman (Summary Quality)

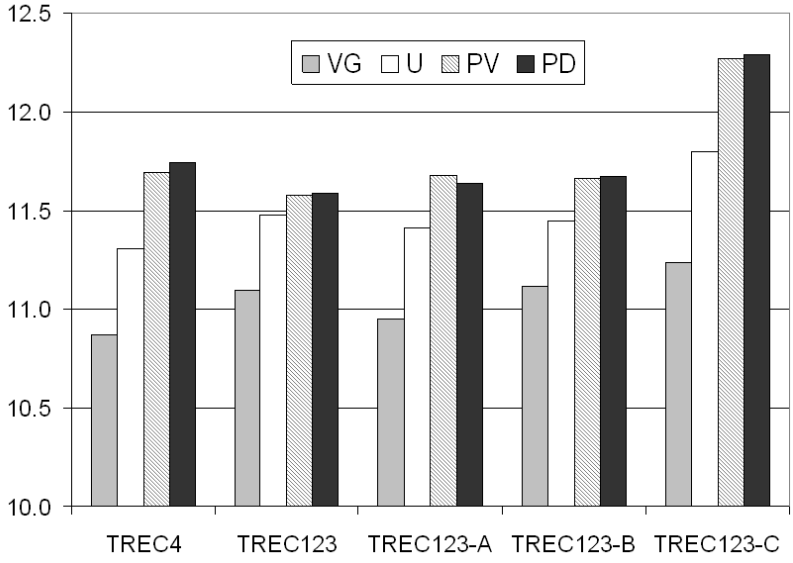

Figure 75: Entropy (Summary Quality)

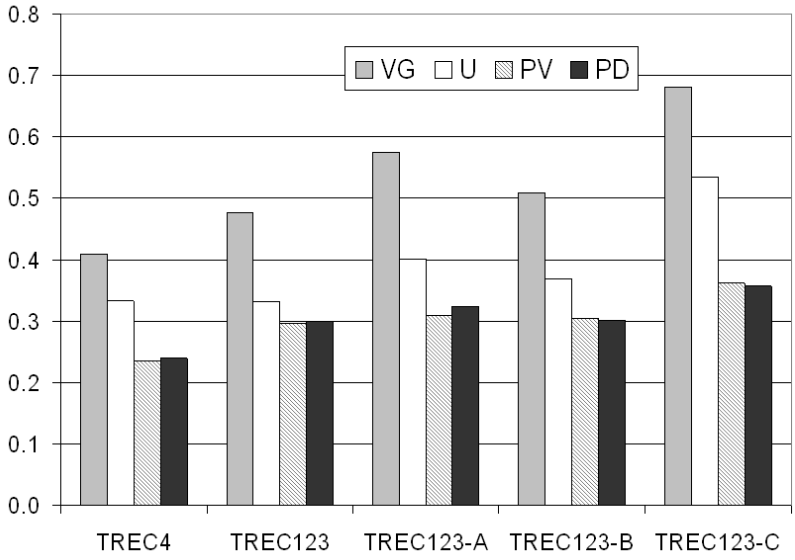

Figure 76: JS-Divergence (Summary Quality) [Lower is Better]

all the schemes tested, since it focuses solely on sampling from the most efficient databases in terms of vocabulary production. What we noticed, though, was that the VG scheme tended to allocate all of the sampling documents to a few small databases each with a fairly large vocabulary. These databases had significantly steep vocabulary growth curves, and as a result, the overall collection vocabulary for the VG approach was higher than for the other approaches. But, since the sampling documents were assigned to only a handful of small databases, the larger databases (which tend to have slower growing vocabulary growth rates) were undersampled. We are interested in further exploring the effectiveness of the VG scheme in application scenarios that rely on rich coverage of vocabulary terms.

6.4.2.3 Additional Results

Given the good results for the PD and PV schemes, we next tweak several of the factors impacting the sampling framework for better understanding of its performance.

First, we consider the impact of the total number of sample documents S on the quality of the extracted database samples. As the framework is able to sample more documents, we would expect to extract higher quality samples. We consider three scenarios. In Scenario 1, we have total sample documents $S = 100 \cdot n$, where n is the number of databases in

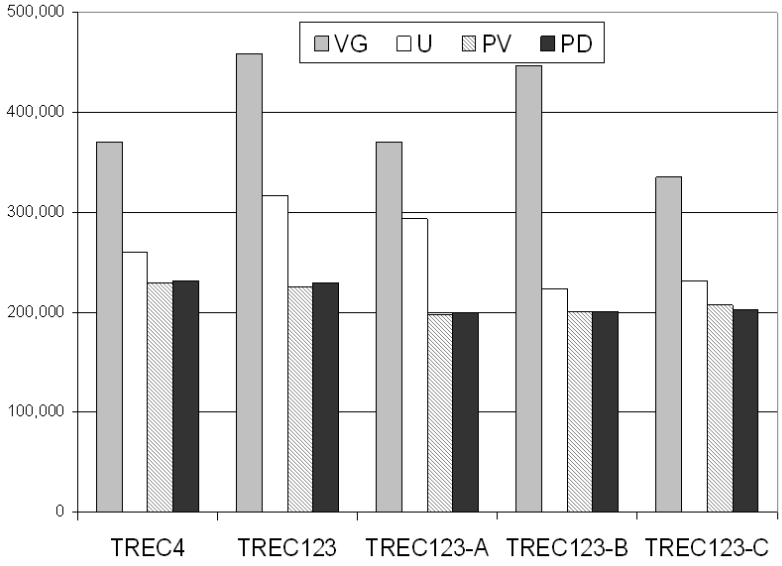

Figure 77: Total Collection Vocabulary Size

the dataset; in Scenario 2, we have $S = 300 \cdot n$; and in Scenario 3, we have $S = 500 \cdot n$. So, for example, the total sampling allocation for TREC123 and its 100 databases is 10,000 documents in Scenario 1, 30,000 documents in Scenario 2, and 50,000 documents in Scenario 3.

In Figures 78 to 80, we show the impact of increasing S over the Uniform sampling approach ($U[100]$, $U[300]$, and $U[500]$) as compared to the Proportional Document sampling scheme ($PD[100]$, $PD[300]$, and $PD[500]$). For the PD cases, we allocate half the documents available for seed sampling (meaning that in Scenario 1, we collect a seed sample of 50 documents from each database; in Scenario 2, we collect a seed sample of 150 documents; in Scenario 3, we collect a seed sample of 250 documents). We restrict Figures 78, 79, and 80 to results for two of the datasets and three of the quality metrics; note that the general results hold for all datasets and quality metrics.

Of course, as we increase the total sample document allocation, both the uniform and quality-conscious sampling schemes result in higher quality samples, since more of each database may be sampled. We note that in all cases, the quality-conscious sampling schemes approach outperforms the uniform approach, even when the total sample document allocation is significantly limited and the sampling framework must rely on even smaller seed

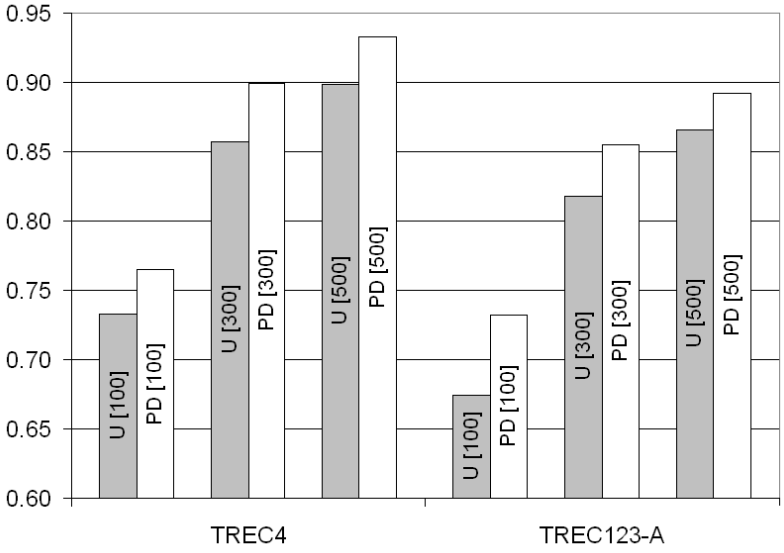

Figure 78: Impact of Increasing the Total Number of Sample Documents S, Weighted Common Terms

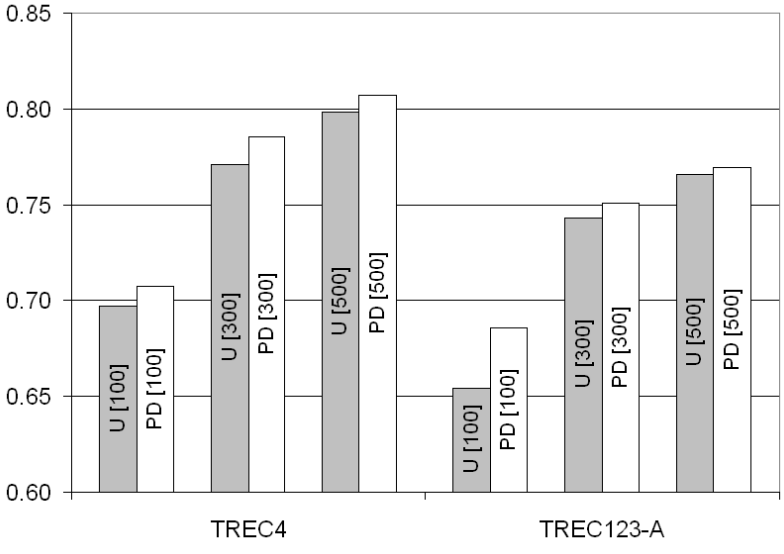

Figure 79: Impact of Increasing the Total Number of Sample Documents S, Spearman

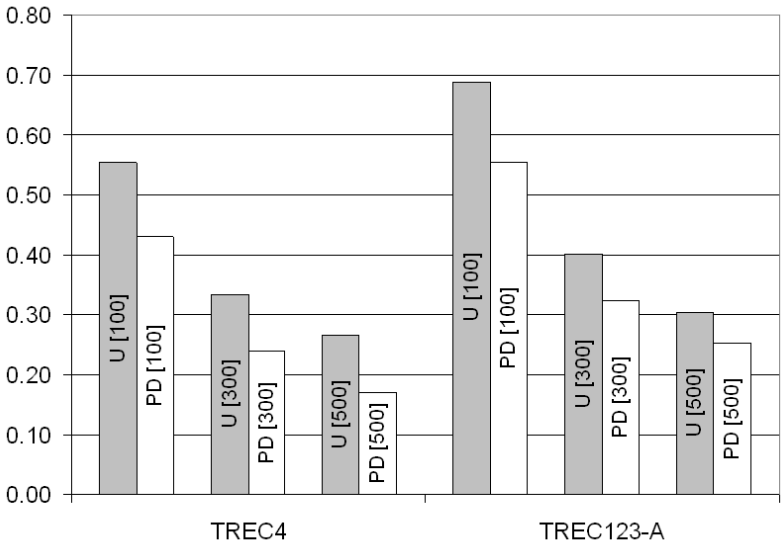

Figure 80: Impact of Increasing the Total Number of Sample Documents S, JS-Divergence [Lower is Better]

samples for estimating the size of the database.

Second, we study the impact of the total allocation to the Seed Sampling step (S_{seed}) versus the dynamic sampling step (S_{dyn}), where recall that $S = S_{seed} + S_{dyn}$. Devoting too many of sampling documents for the seed sampling may result in more precise estimates for use by each quality-conscious sampling scheme, but leave too few documents available for dynamic sampling. Conversely, devoting too few sampling documents for the seed sampling may result in less precise parameter estimates, and hence may lead to the sampling scheme misallocating the remaining documents for the dynamic sampling step.

In Figures 81 to 83, we show the impact of the total seed sampling allocation for the Proportional Document scheme with $S = 300 \cdot n$, where n is the number of databases in the dataset. We consider three scenarios – in Scenario 1, we collect a seed sample of 50 documents from each database ($PD[50, 250]$), leaving $250 \cdot n$ documents available for dynamic sampling; in Scenario 2, we collect a seed sample of 150 documents from each database, leaving $150 \cdot n$ ($PD[150, 150]$); in Scenario 3, we collect a seed sample of 250 documents, leaving only $50 \cdot n$ documents available for dynamic sampling ($PD[250, 50]$). For comparison, we also consider the uniform sampling approach U. We restrict Figures 81,

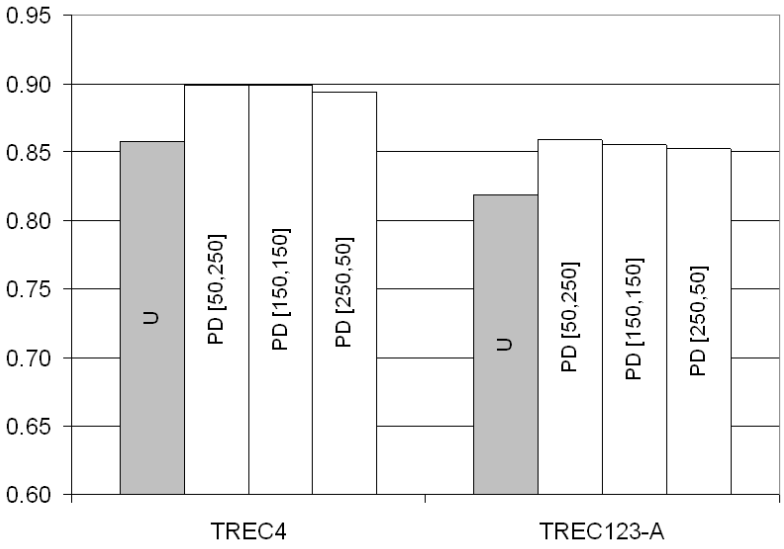

Figure 81: Impact of Seed Sampling, Weighted Common Terms

82, and 83 to results for two of the datasets and three of the quality metrics; note that the general results hold for all datasets and quality metrics.

Interestingly, the dynamic sampling approach results in higher quality samples in all cases. As the seed sampling allocation of documents increases, the advantage of the quality-conscious sampling schemes is diminished only slightly relative to the uniform approach. This is expected since fewer documents are available for dynamic sampling, and hence the performance will converge to the uniform approach.

Finally, we have also studied the impact of the number of rounds on the multi-round distributed query-sampling framework. All of our experiments so far have relied on a single round of dynamic sampling. Interestingly, we find that increasing the number of rounds from 1 to 10 results in slight improvements to the extracted database samples primarily due to the refined parameter estimates made possible by re-calculating the appropriate allocation after each round.

6.4.3 Application Scenario: Database Selection

We have shown that the adaptive query-sampling framework results in higher-quality database samples compared to the uniform sampling approach used in many existing query sampling

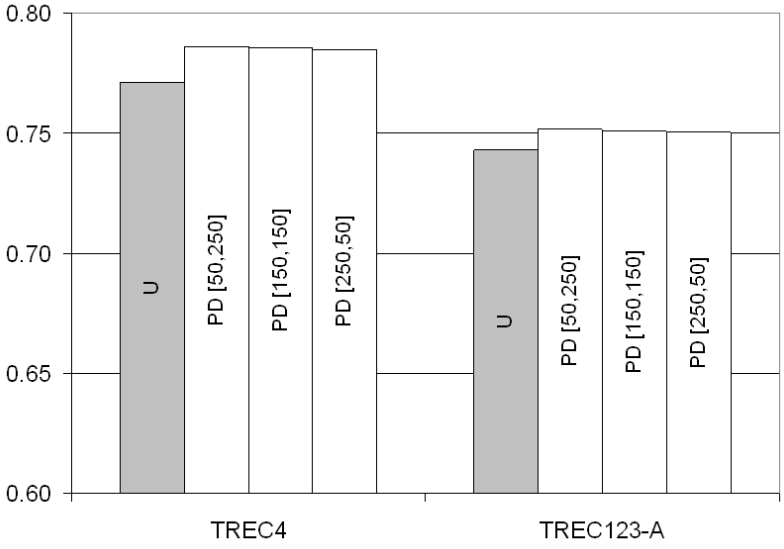

Figure 82: Impact of Seed Sampling, Spearman

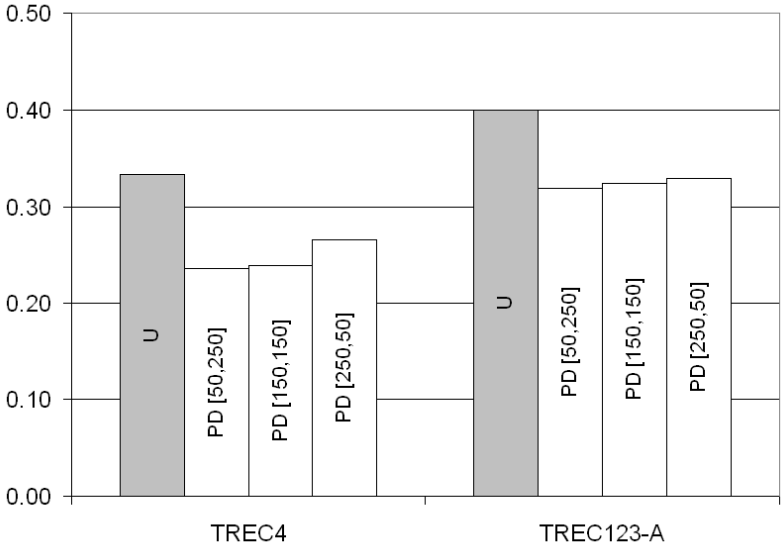

Figure 83: Impact of Seed Sampling, JS-Divergence [Lower is Better]

schemes. In this section, we evaluate the impact of our adaptive sampling framework on the real-world application of database selection. Current approaches for text database selection map queries to databases based on previously acquired metadata for each database. Due to network bandwidth and time constraints, the goal of database selection is to identify only those databases that can contribute to a particular query by analyzing the metadata of each database.

Typical database selection algorithms work in the context of a query q and a set of candidate databases $\mathcal{U}$. For each database $D \in \mathcal{U}$, a goodness (or quality) score is assigned in terms of query relevance. The databases are ranked according to the relevance score and the query is then routed to the top-k ranked databases. In our experiments, we consider the popular CORI algorithm as introduced in [34] and described in [63]. CORI assigns scores according to the function: $CORI(q, D) = \sum_{t \in q} \frac{0.4+0.6TI}{|q|}$, where $T = \frac{c(t,D_s) \cdot |D_s|}{c(w,D_s) \cdot |D_s| + 50 + 150 \cdot \frac{f(D)}{f(D)}}$ and $I = \frac{\log(\frac{n+0.5}{c(t,U)})}{\log(n+1)}$, where $c(t, U)$ = number of dbs containing t, and $f(\bar{D}) = \frac{1}{n} \sum_{i=1}^{n} f(D_i)$ = mean number of words among dbs being ranked. Obviously the quality of such a database selection algorithm will be impacted by the quality of the frequency and count estimates generated by the document samples from the sampling process.

For the TREC123-A, B, and C datasets, we use queries drawn from the TREC topics 51-100 title field. These queries are, on average fairly short (ranging from 1 to 11 words, with an average of 3.8), and resemble Web-style short keyword queries. For the TREC4 dataset, we use queries from the TREC topics 201-250 description field (ranging from 8 to 33 words, with an average of 16 words).

To isolate the quality of database selection from the rest of the distributed information retrieval problem (which also includes components for results merging and ranking), we adopt a commonly accepted criterion for measuring the database selection recall. The database selection recall metric, denoted by R_n, evaluates the quality of the database selection algorithm's ranked list of databases versus a baseline ranking [72]. Several baseline rankings have been proposed before, including (1) size-based ranking – a query-independent ranking that ranks databases in descending order of size; (2) relevance-based ranking – a query-dependent ranking that ranks databases by the number of query-relevant documents

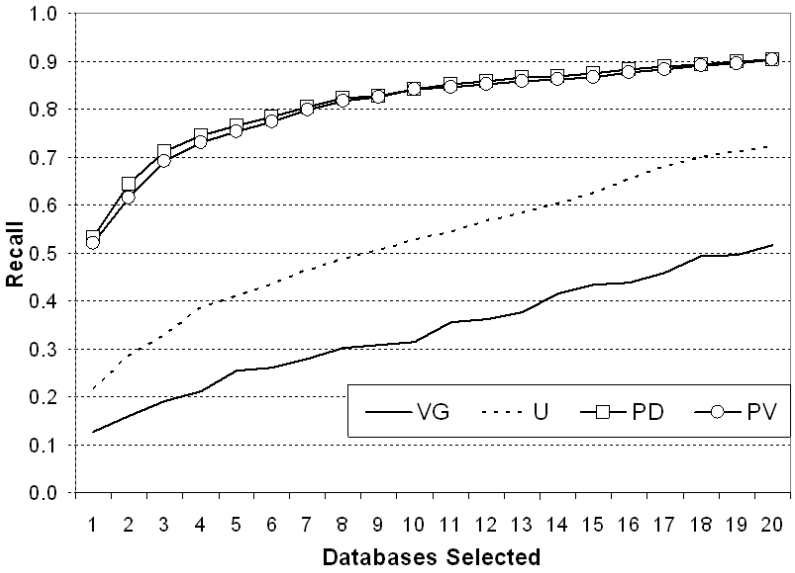

Figure 84: Database Selection Recall, TREC123-A

each has; and (3) best-document ranking – where databases are ranked by the similarity of the best document to the query [123]. In this chapter, we rely on the commonly used relevance-based ranking, partly because the TREC data comes with relevance decisions for several query sets. If we let $rel(q, D)$ denote the number of relevant documents in database D to the query q, then for a baseline ranking of n databases: $B = (B_1, ..., B_n)$ and a ranking induced by the database selection algorithm $E = (E_1, ..., E_n)$, we may define the recall for a particular query q as: $R_k(q) = \frac{\sum_{i=1}^{k} rel(q,E_i)}{\sum_{i=1}^{k} rel(q,B_i)}$, where $0 \leq R_k(q) \leq 1$. By evaluating $R_k(q)$ for different values of k, we may assess the recall at different levels (e.g., recall for the top-5 databases, the top-10, and so on). A database selection algorithm that induces a ranking that exactly matches the baseline (optimal) ranking, will result in R_k values of 1 for all choices of k.

Finally, we run experiments on the database samples using the setup described in Section 6.4.2.2 to compare our adaptive sampling framework with the uniform sample allocation scheme. For each dataset, we evaluated the CORI algorithm over the extracted database samples and the query mix discussed above. In Figures 84 to 86, we report the database recall metric for the TREC123-A, B, and C datasets. These results confirm that the higher quality PV and PD samples reported in the earlier set of experiments positively impact

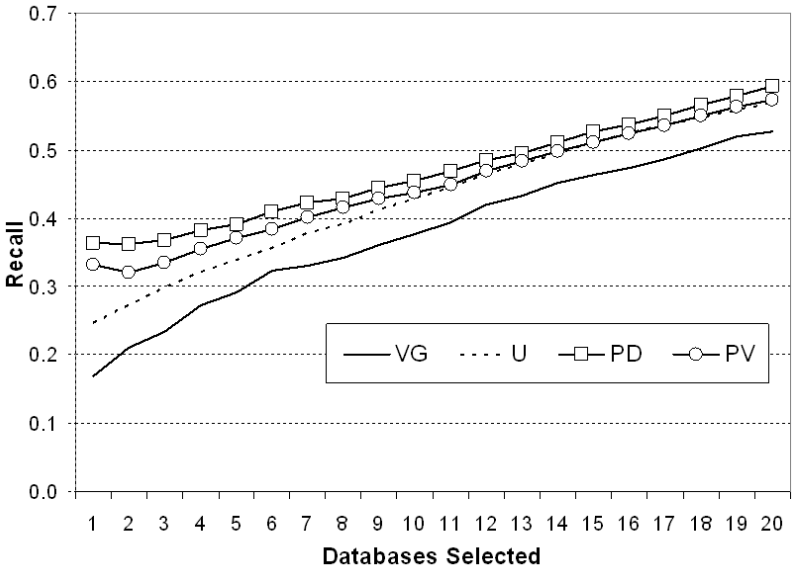

Figure 85: Database Selection Recall, TREC123-B

the performance of database selection relative to the uniform approach, and again, the VG scheme significantly lags. We see a similar, though less pronounced, advantage over the TREC4 and TREC123 datasets, as shown in Figures 88 and 87.

6.5 Related Work

The distributed query-sampling sampling framework presented in this chapter is designed to extract high-quality database samples from distributed text databases like those on the Deep Web.

The Deep Web (sometimes referred to as the hidden or invisible Web) has garnered increased research interest in recent years. Several studies have noted the immense size of the Deep Web relative to the surface Web of static documents [16, 39, 117]. One of the first Deep Web crawlers for discovering and interacting with Deep Web databases was proposed in [145], where the complexity of interacting with Web search interfaces was noted. More recently, there have been efforts to match Deep Web query interfaces [169, 182, 193], and a study of how to efficiently sample an entire collection [135].

The sampling-based approach at the heart of the framework presented in this chapter

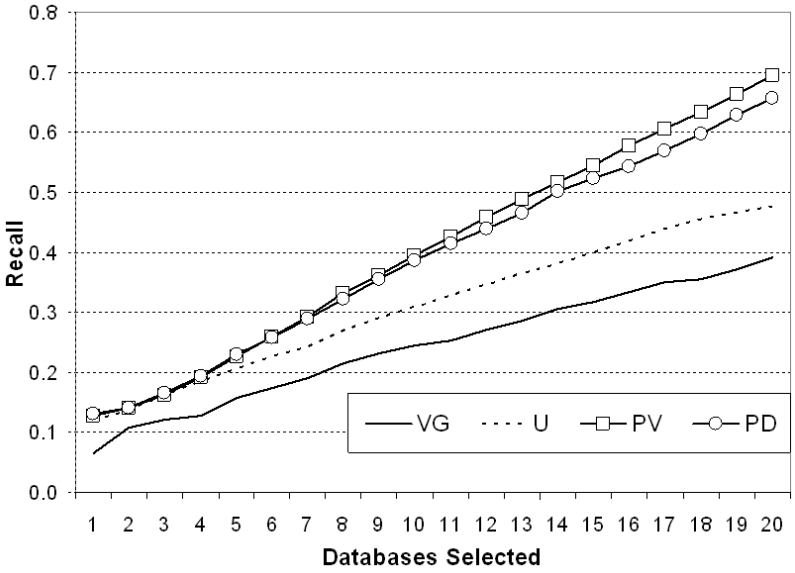

Figure 86: Database Selection Recall, TREC123-C

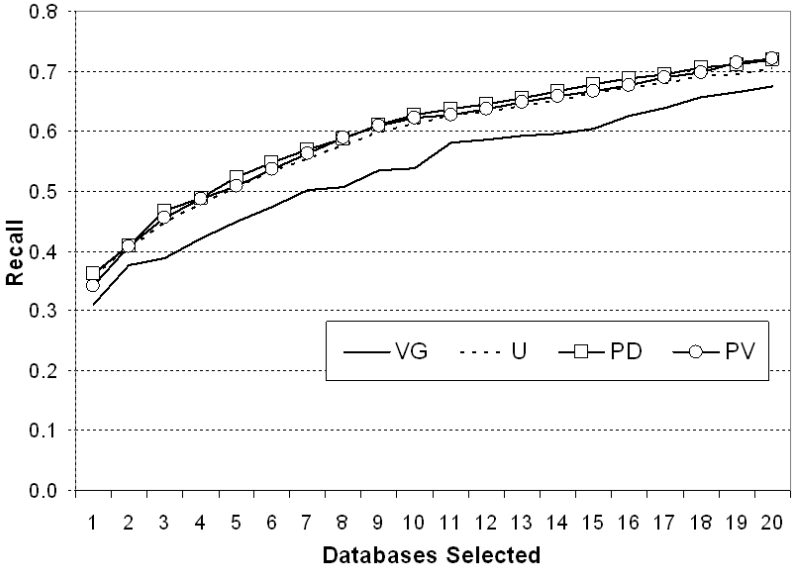

Figure 87: Database Selection Recall, TREC4

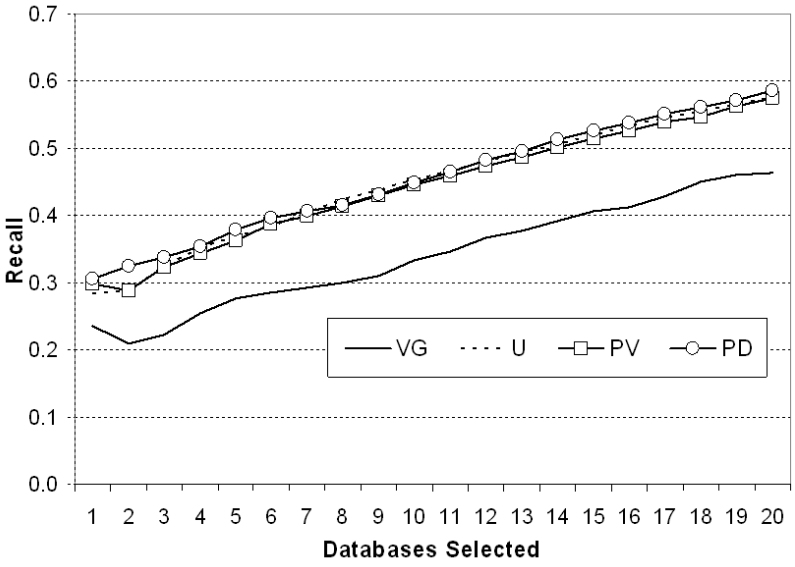

Figure 88: Database Selection Recall, TREC123

has garnered previous research attention and has shown some success in many contexts, including distributed information retrieval, database selection, database categorization, peer-to-peer information retrieval, and others, e.g., [35, 91, 114, 160].

There have been several studies on how to *sample* a database in an effort to generate a summary of the database internals [36, 37, 43, 74, 84, 91, 93, 124, 166, 170]. The main purpose of these techniques is to generate a representative content summary of the underlying database in cases where the database provides only limited access. These sampling approaches typically rely on interacting with the database through a query interface and extracting sample data through a series of query probes. Querying methods suggested include the use of random queries, queries learned from a classifier, and queries based on a feedback cycle between the query and the response. In addition, Agichtein et al. [4] have developed a formal reachability graph model for assessing the quality of query-based database summarization techniques. A common assumption in the literature is to extract 300 documents from a database to create a summary, e.g., [90, 133, 161]. In contrast to the sampling-based approach, other researchers have studied the problem of downloading the entire contents of a Deep Web site using only a query-based mechanism for extracting documents [135].

6.6 Summary

We have presented the adaptive *distributed query-sampling framework* for optimizing the quality of database samples. The optimization is performed by dynamically revising the sampling allocation among the n databases based on estimated quality metrics derived *during* the sampling process. To the best of our knowledge, our framework is the first one for sampling that takes into account both the overall quality of all text databases under consideration *and* the presence of realistic resource constraints. We have introduced three concrete sample allocation schemes for estimating database quality, and have shown how the adaptive framework supports higher-quality document sampling than the existing solutions, and how the real world application of database selection may be improved.

CHAPTER VII

TRUSTED WEB RESOURCE DISCOVERY

In the previous chapter, we introduced the controlled sampling approach for extracting high-quality Web resource samples to reduce the opportunities of attackers to corrupt Web-based categorization and integration services. In this chapter, we use the controlled sampling approach as a foundation for supporting trusted Web resource discovery. By leveraging a set of user-trusted information resources, the system provides a set of related information resources that can be grouped by content and trustworthiness properties to auto-generate a trusted categorization hierarchy.

7.1 Overview

Our research interest is to provide support for the management of Deep Web databases beyond that captured by traditional search engines. Typically, a search engine is optimized to identify a ranked list of documents (or Web pages) relevant to a user query. This *document-centric* view has proven immensely successful. On the other hand, with the rise of high-quality Deep Web databases and the emergence of digital libraries and corporate information servers, we believe that there is ample opportunity for a new class of queries optimized not on the document level, but on the more general relationship level between Deep Web databases.

Rather than requesting the top-ranked documents containing a certain keyword, say "autism", we propose that a user may be more interested in *relationship-centric* queries about the many available Deep Web databases. For example, a user familiar with the popular online medical literature site PubMed[1] that provides access to millions of scientific and medical literature citations may be interested in posing some of the following queries:

- What other Deep Web databases are most similar to PubMed?

[1] `http://www.ncbi.nlm.nih.gov/PubMed/`

- What other Deep Web databases are more general than PubMed? Or more specialized?

- Are there any other Deep Web databases that complement PubMed's coverage?

We could also imagine extending these queries to more sophisticated ones that cover relationships among multiple Deep Web databases. Additionally, the granularity of the relationship discovered may be further refined to consider subsets of the databases. For example, a user may be interested in relationship-centric queries about specific journals within the PubMed database, and not PubMed as a whole.

We envision a number of scenarios in which a relationship-centric framework over Deep Web databases would be of significance:

- First, a relationship-centric framework can be used for supporting direct relationship queries about Deep Web databases like the ones listed above.

- In addition to these types of direct relationship queries, a user may simply be interested in discovering any non-obvious relationships that may exist among a group of Deep Web databases. The relationship-centric framework supports this type of data mining.

- The relationship-centric framework may also be used to augment traditional document-centric queries over Deep Web databases. For example, a user interested in medical literature may choose to query both PubMed and all databases with a similarity-based relationship to PubMed. Alternatively, a user interested in maximizing coverage of multiple topically-distinct Deep Web databases, may choose to query both PubMed and any other Deep Web database that has complementary coverage relative to PubMed.

- Finally, the relationship-centric framework may also support the refinement and generalization of traditional document-centric queries over Deep Web databases. A user issuing a query to PubMed may be overwhelmed by responses from a number of different medical journals. She may prefer to refine the query to the Deep Web databases

that are more specialized on a particular topic like cancer research. Alternatively, she may find too few responses to a particular query and wish to generalize the scope of the query to include more Deep Web databases that have broader coverage than PubMed.

Currently, there are no effective means to answer queries that rely on a relationship-centric view of Deep Web databases without relying on significant human intervention or hand-tuned categorization schemes. Search engines rely on a document-centric view of the Web and are not designed to handle relationship-level queries. Similarly, directory services – like the ones offered by Yahoo! and the Open Directory Project [dmoz.org] – offer general categories but do not provide coverage and specialty ratings for direct comparisons between Deep Web databases. So a user may find a category listing for medical literature resources that includes PubMed, but she would lack support for understanding the relationship between PubMed and the other medical literature resources, or for understanding the relationship between PubMed and resources listed under a different category.

With the rapid increase in the number and variety of available Deep Web databases, there is a growing need for providing a relationship-centric framework for discovering and understanding the interrelationships among Deep Web databases. To answer these challenges, we present a novel approach to discover interesting relationships among Deep Web databases based on *source-biased database analysis*. This source-biased approach supports a relationship-centric view over a collection of Deep Web databases through source-biased probing and source-biased relevance metrics. Source-biased database analysis and discovery presents a number of unique properties. First, our approach is capable of answering relationship-centric queries of the form posed above by focusing on the nature and degree of the relationship of one Deep Web database to another. Since Deep Web databases tend to be large and may be updated frequently ([92, 117]), we rely on a *sampling-based* approach to extract a portion of each Deep Web database through source-biased probing. Given a database like PubMed – called the *source* - the source-biased probing technique leverages the summary information of the source to generate a series of biased probes to other databases – called the *targets*. This source-biased probing allows us to determine whether a target

database is relevant to the source by probing the target with very few focused probes. Second, we introduce the *biased focus* metric to discover highly relevant Deep Web databases and to assess the nature of the relationship between databases. For each source database, we use the biased focus to rank target databases and to identify interesting relationship sets that support relationship-centric queries. Such relationships can be further utilized as value-added metadata for each database discovered. Third, to further reduce the number of probes necessary to assess a target database, we introduce a performance optimization called *source-biased probing with focal terms*. The main idea is to identify terms in the unbiased summary of the source that share a similar topical category and then to divide the source summary into k clusters, where each cluster represents a group of similar summary terms.

Our experiments on both simulation and Web datasets show how the source-biased database analysis approach results in efficient discovery and ranking of Deep Web databases. We also illustrate how our approach supports relationship-centric queries through the identification of interesting relationship sets, including similarity-based and hierarchical relationship sets. Additionally, we present the design and architecture of our DynaBot system for crawling, probing, and supporting relationship-centric queries over Deep Web databases using the source-biased approach.

The rest of the chapter is organized as follows. We present related work in Section 7.2 and briefly discuss the motivation and system model in Section 7.3. In Section 7.4, we describe the algorithm for source-biased analysis of databases, including source-biased probing, the biased-focus metric, and the assessment of inter-database relationships. We refine the basic source-biased algorithm with focal terms in Section 7.5. We present the design and architecture of the DynaBot crawler for supporting relationship-centric queries over the Deep Web in Section 7.6. In Section 7.7, we provide extensive experimental evidence to demonstrate the effectiveness of our algorithms and end in Section 7.8 with our final thoughts.

7.2 Related Work

In the database community, considerable attention has been dedicated to the *database selection* problem [34, 46, 55, 63, 65, 72, 73, 84, 111, 123, 141, 191]. In database selection, the problem is to take a query and match it to potentially relevant databases for processing. Typically the database exports a description to help guide database selection. Instead of matching a query to a set of databases, our work is concerned with analyzing and understanding the relationships among Deep Web databases in a source-biased framework. As we will show, these interesting relationships may be used to help guide database selection in addition to supporting relationship-centric queries.

As we discussed in Chapter 6, other researchers have previously studied the problem of *sampling* a database in an effort to generate a summary of the database internals [36, 37, 43, 74, 84, 91, 93, 124, 166, 170]. In this chapter, we show how traditional unbiased sampling approaches – especially in the context of the large size and dynamic nature of Deep Web databases – may be inadequate for exposing relationships among Deep Web databases. As a result, we promote a source-biased perspective to overcome these issues.

Others [74, 93] have introduced a probing-based approach for classifying Deep Web databases into a pre-determined Yahoo!-style hierarchy. However, this approach relies on a pre-learned set of queries for database classification, which requires the potentially burdensome and inflexible task of labelling training data for learning the classifier probes in the first place. Additionally, if new categories are added or old categories removed from the hierarchy, new probes must be learned and each source re-probed. Our approach is designed to work flexibly in a "bottom-up" fashion by placing each Deep Web database at the center of its own neighborhood of related databases.

7.3 System Model and Problem Statement

In this section, we present the system model and discuss in detail current approaches for sampling Deep Web databases. We identify problems with the current approaches that motivate the source-biased framework for identifying interesting relationships among Deep Web databases. The large and increasing number of Deep Web databases accessible through

the Web not only makes a relationship-centric view over a collection of Deep Web databases important but also demands for an efficient and effective framework for discovering and understanding the interesting interrelationships among Deep Web databases.

7.3.1 Modeling Deep Web Databases

We consider a *Web-enabled document database* (or *Deep Web database*) to be a database that is composed primarily of text documents and that provides query-based access to these documents either through keyword search or more advanced search operators. In particular, we are interested in Deep Web databases that are beyond the control of the typical users. For example, Web data sources that provide a search mechanism like the resources on the Deep Web, digital libraries, and databases deployed across loose corporate federations are all examples of Deep Web databases for which a typical user can only access through query-based mechanisms. For such types of Deep Web databases, relationship-centric queries of the form: "What other Deep Web databases are most similar to X? Or complementary to X?" would require significant human intervention to yield a satisfactory result.

We consider a universe of discourse $\mathcal{U}$ consisting of d Deep Web databases: $\mathcal{U} = \{D_1, D_2, \ldots, D_d\}$ where each database produces a set of documents in response to a particular query. Hence, we describe each Deep Web database D_i as a set of M_i documents: $D_i = \{doc_1, doc_2, \cdots, doc_{M_i}\}$. There are N terms (t_1, t_2, ..., t_N) in the universe of discourse $\mathcal{U}$, where common stopwords (like "a", "the", and so on) have been eliminated. Optionally, the set of N terms may be further refined by stemming [140] to remove prefixes and suffixes.

Adopting a vector-space model [154] of the database contents, we may describe each Deep Web database D_i as a vector consisting of the terms in the database along with a corresponding weight:

$$\textsc{Summary}(D_i) = \{(t_1, w_{i1}), (t_2, w_{i2}), \cdots, (t_N, w_{iN})\}$$

A term that does not occur in any documents in Deep Web database D_i will have weight 0. Typically, for any particular Deep Web database D_i, only a fraction of the N terms will have non-zero weight. We refer to the number of non-zero weighted terms in D_i as N_i.

This vector-based approach has received great attention and support in the information retrieval community for representing documents where it has been applied effectively across a wide variety of application settings [13]. In the database community, the vector-based approach has been popularized by GlOSS [72] and related projects.

We call the vector SUMMARY(D_i) a *resource summary* for the Deep Web database D_i. A resource summary is a single aggregate vector that summarizes the overall distribution of terms in the set of documents produced by the database. To find SUMMARY(D_i), we must first represent each document doc_j ($1 \leq j \leq M_i$) as a vector of terms and the frequency of each term in the document:

$$doc_j = \{(t_1, freq_{j1}), (t_2, freq_{j2}), \cdots, (t_N, freq_{jN})\}$$

where $freq_{jk}$ is the frequency of occurrence of term t_k in document j. The initial weight for each term may be based on the raw frequency of the term in the document and it can be refined using alternative occurrence-based metrics like the normalized frequency of the term and the term-frequency inverse document-frequency (*TFIDF*) weight. TFIDF weights the terms in each document vector based on the characteristics of all documents in the set of documents.

Given a particular encoding for each document, we may generate the overall resource summary for each Deep Web database in a number of ways. Initially, the weight for each term in the resource summary may be based on the overall frequency of the term across all the documents in the database (called the *database frequency*, or *dbFreq*):

$$w_{ik} = dbFreq_{ik} = \sum_{j=1}^{M} freq_{jk}$$

Alternatively, we can also define the weight for each term based on the number of documents in which each term occurs (called the *document count frequency*, or *docCount*):

$$w_{ik} = docCount_{ik} = \sum_{j=1}^{M} \mathcal{I}_j(t_k)$$

where $\mathcal{I}_j(t_k)$ is an indicator function with value 1 if term t_k is in document j and 0 otherwise.

Once we have chosen our database model, to effectively compare two Deep Web databases and determine the relevance of one database to another, we need two technical components: (1) a technique for generating a resource summary; and (2) a metric for measuring the relevance between the two databases.

7.3.2 Estimating Resource Summaries

Ideally, we would have access to the complete set of documents belonging to a Deep Web database. We call a resource summary for D_i built on the complete set of documents an *actual resource summary* or $\text{ASUMMARY}(D_i)$. However, the enormous size of many Deep Web databases coupled with the non-trivial costs of collecting documents (through queries and individual document downloads) make it unreasonable to generate an actual resource summary for every Deep Web database available. Additionally, the well-noted dynamic nature of Deep Web data [92] makes extracting the complete set of documents belonging to a Deep Web database infeasible, since Deep Web databases may add new content and delete old content faster than all documents may be extracted.

As a result, previous researchers have introduced several techniques for *sampling* a database through a series of *probe queries* to generate a representative summary based on a small sample of the entire database [37, 36]. We call such a representative summary an *estimated resource summary*, denoted as

$$\text{ESUMMARY}(D_i) = \{(t_1, w_{i1}), (t_2, w_{i2}), \cdots, (t_N, w_{iN})$$

The number of occurring terms (i.e., those terms that have non-zero weight) in the estimated summary is denoted by N_i'. Typically, N_i' will be much less than the number of non-zero weighted terms N_i in the actual resource summary since only a fraction of the total documents in a database will be examined. Hence, the goal of a prober is typically to find $\text{ESUMMARY}(D_i)$ such that the relative distribution of terms closely matches the distribution of terms in $\text{ASUMMARY}(D_i)$, even though only a fraction of the total documents will be examined.

Current probing techniques for estimating resource summaries aim at estimating the

overall summary of the content for a Deep Web database. We classify these sampling techniques into two categories: random sampling and query-based sampling.

Random Sampling – No Bias

If we had unfettered access to a Deep Web database, we could randomly select terms from the database to generate the estimated resource summary ESUMMARY(D_i). Barring that, we could randomly select documents with which to base the estimated resource summary. We will call such a random selection mechanism an *unbiased prober* since all terms (or documents) are equally likely to be selected. In practice, an unbiased prober is unrealistic since most Deep Web databases only provide a query-based mechanism for extracting documents.

Query-based Sampling – Query Bias

As a good approximation to unbiased probing, Callan et al. [37, 36] have introduced a query-based sampling technique for generating accurate estimates of Deep Web databases by examining only a fraction of the total documents. The Callan technique relies on repeatedly requesting documents from a source using a limited set of queries. Since the documents extracted are not chosen randomly, but are biased by the querying mechanism through the ranking of returned documents and by providing incomplete access to the entire database, we say that the Callan technique displays *query bias*. There are several ways to define the limited set of queries, including random selection from a general dictionary and random selection augmented by terms drawn from the extracted documents from a database. The query bias technique has some advantages over the no bias approach, since two databases that consist of the same set of documents, but have different ranking approaches will be sampled (and hence represented by the estimated summary) differently. Through experimental validation, the Callan technique has been shown to extract high-quality resource summaries (through a number of metrics) over databases consisting of over one million pages by collecting only a few hundred documents. Since the distribution of terms across the documents of a text database follows a Zipfian distribution [36] – meaning that a few words occur in many documents, while the majority of terms occur in very few documents – extracting a small sample of documents may be successful at extracting the most popular

terms in a database, resulting in a high-quality resource summary. In the rest of the chapter, when we refer to an estimated resource summary ESUMMARY(D_i), we mean one that has been produced by a query-biased prober.

7.3.3 Potential Problems

In order to determine the relevance of one Deep Web database D_i to another database D_j and to assess the nature of their relationship, we require an appropriate relevance metric. There are a number of possible relevance metrics to compare two resource summaries, including a simple count of the common terms in both resource summaries to a weighted version that considers the term weights of the common terms in both resource summaries. We consider two different relevance metrics here that each emphasize a different notion of relevance.

The first is a popular symmetric relevance metric adopted from the information retrieval community for comparing the resource summaries of two databases D_i and D_j – the cosine similarity (or normalized inner product):

$$cos(D_i, D_j) = \frac{\sum_{k=1}^{N} w_{ik} w_{jk}}{\sqrt{\sum_{k=1}^{N} (w_{ik})^2} \cdot \sqrt{\sum_{k=1}^{N} (w_{jk})^2}}$$

where w_{ik} is the weight for term k in ESUMMARY(D_i) and w_{jk} is the weight for term k in ESUMMARY$_{D_j}$. The cosine ranges from 0 to 1, with higher scores indicating a higher degree of similarity. In contrast, the cosine between orthogonal vectors is 0, indicating that they are completely dissimilar. The cosine measures the angle between two vectors, regardless of the length of each vector.

The second relevance metric is asymmetric in nature and measures the fraction of terms in one resource summary that are also contained in the other resource summary:

$$contain(D_i, D_j) = \frac{|\text{ESUMMARY}(D_i) \cap \text{ESUMMARY}(D_j)|}{|\text{ESUMMARY}(D_i)|}$$

If all of the terms in D_i also occur in D_j, then the containment is 1. If no terms in D_i occur in D_j then the containment is 0. The containment relevance metric may be used

to measure the containment of each resource summary with respect to the other, and vice versa.

We now use an example to illustrate why the existing resource summary estimation techniques are inadequate for effectively revealing interesting relationships between two databases, especially in terms of the content coverage of one (target) in the context of the other (source). We considered three real-world Deep Web databases – the PubMed medical literature site, the job-posting site Monster.com, and the popular search engine Google. We consider Google to be a Deep Web database since it provides an advanced ranking engine over a database of indexed content. We further note that all three Deep Web databases provide access to vast content that is constantly being updated (i.e., PubMed adds new citations, Monster provides up-to-date job postings, and Google updates its index to reflect changes in the Web), meaning that a sampling-based approach is necessary to provide a current snapshot of the state of each Deep Web database.

Example: *We collected 300 documents from Google, PubMed, and Monster, respectively, using a query-based sampling technique for resource summary estimation. For each site, we issued a random query term drawn from the standard Unix dictionary and collected a maximum of four documents per query. We then extracted the plain text from each sampled document by removing all HTML tags and eliminated a list of common stopwords (e.g., "a", "the", and so on). Using the resource summaries constructed, we find that* $cos(PubMed, Google) = 0.15$ *and* $cos(PubMed, Monster) = 0.16$*. Similarly, we find that for the asymmetric relevance metric, we have* $contain(PubMed, Google) = 0.19$ *and* $contain(PubMed, Monster) = 0.22$*, meaning that 22% of the terms in the PubMed estimated summary also occur in the Google estimated summary, whereas 25% of the PubMed terms occur in Monster. Hence, for both relevance metrics we see that both Google and Monster appear to have relatively low relevance with respect to PubMed. Although Monster does provide some health-related content (like medical job postings), we expect that Google should be significantly more relevant to PubMed since it provides as much or more health-related content. When we reverse the containment calculation, we find that* $contain(Google, PubMed) = 0.32$ *and* $contain(Monster, PubMed) = 0.36$*. Interestingly, these statistics support exactly*

the opposite conclusion we would anticipate: that more of Google's content is contained in PubMed, than vice versa.

This example underlines a critical problem with current techniques for probing and comparing resource summaries. Current resource summary estimation techniques are concerned with generating *overall* summaries of the underlying databases. The goal is to generate essentially an unbiased estimate of the actual resource summary. Due to the Zipfian distribution of terms across the documents of a Deep Web database, an unbiased summary will focus on terms that are relatively popular across the space of all documents. For two Deep Web databases that have overlapping content (like PubMed and Google), a comparison over these unbiased summaries will not necessarily identify these specialized areas of commonality. In this example, since Google has such broad coverage, few terms in an unbiased estimated summary may be common to the PubMed estimated resource summary. Hence, many topics that are relevant for context-based database comparisons may be under-represented or overlooked completely, since the summaries contain just a small fraction of the total terms in each database. In the context of our example, it would be interesting to discover both that Google is much more relevant to PubMed than Monster and that Google has much broader coverage than Monster. When comparing databases, the relevance metrics based on the unbiased summaries of each database like the ones described above are clearly inadequate, since they fail to adequately capture the asymmetric relationship between the two databases. This example shows both the need for a new query probing technique to hone on the common areas between two databases, allowing for more in-depth comparisons, and the need for more effective relevance metrics to more appropriately capture the nature and degree of the relationship between two databases.

7.4 Source-Biased Database Analysis

Bearing these issues in mind, we introduce a source-biased approach – called *source-biased database analysis* – to efficiently discover interesting relationships among text document databases. There are three fundamental steps in performing source-biased database analysis: (1) source-biased probing for Deep Web database discovery; (2) evaluation and ranking

of discovered Deep Web databases with the biased focus metric; and (3) leveraging the biased perspective of sources and targets to discover interesting relationships. This framework provides the foundation for enabling relationship-centric queries over Deep Web databases.

7.4.1 Source-Biased Probing

In order to find the target databases that have high relevance to a source, we need to generate a source-biased summary of the target databases instead of using unbiased summaries of the targets. We introduce a source-biased probing algorithm that can compute the relevance of the target databases with respect to the source in very few probes. Given a Deep Web database – the *source* – the source-biased probing technique leverages the summary information of the source to generate a series of biased probes for analyzing another Deep Web database – the *target*. This source-biased probing allows us to determine in very few interactions whether a target database is relevant to the source by probing the target with focused probes. Note that the goal of source-biased probing is *not* to generate an unbiased estimated resource summary like the query-based sampling approach discussed above. Instead, the goal is to intentionally skew the estimated summary of a target Deep Web database towards the source database for enabling more effective comparisons. This source-biasing is especially important for supporting relationship-centric queries over a diverse and large set of Deep Web databases for which exhaustive sampling of each database is infeasible.

To help differentiate the source-biased approach from others discussed in Section 7.3, in this section we use σ to denote the source database and τ to denote the target database instead of D_i and D_j. Given two databases σ and τ, the output of the source-biased probing is a subjective resource summary for τ that is biased towards σ. We denote the source-biased summary of the target database as:

$$\text{ESUMMARY}_{\sigma}(\tau) = \{(t_1, w_1^{\sigma}), (t_2, w_2^{\sigma}), \cdots, (t_N, w_N^{\sigma})\}$$

N is the total number of terms used in analyzing the set of Deep Web databases. w_i^{σ} $(1 \leq i \leq N)$ is the weight of term t_i, defined using one of the weight functions introduced

SourceBiasedProbing(Database σ, Database τ)
For target database τ, initialize $\text{ESUMMARY}_\sigma(\tau) = \emptyset$.
repeat
Invoke the probe term selection algorithm to select a one-term query
probe q from the source of bias $\text{ESUMMARY}(\sigma)$.
Send the query q to the target database τ.
Retrieve the top-m documents from τ.
Update $\text{ESUMMARY}_\sigma(\tau)$ with the terms and frequencies from the top-m documents.
until Stop probing condition is met.
return $\text{ESUMMARY}_\sigma(\tau)$

Figure 89: Source-Biased Probing Algorithm

in Section 7.3.1. It is important to note that typically the inequality $w_j \neq w_j^\sigma$ does hold.

Concretely, the source-biased probing algorithm generates a source-biased summary for a target as follows: It begins with an unbiased resource summary of the source σ, denoted by $\text{ESUMMARY}(\sigma)$, that is generated through a standard application of query-based sampling. It uses the unbiased resource summary $\text{ESUMMARY}(\sigma)$ as a dictionary of candidate probe terms and sends a series of probe terms, selected from $\text{ESUMMARY}(\sigma)$, to the target database τ; for each probe term, it retrieves the top m matched documents from τ, generates summary terms and updates $\text{ESUMMARY}_\sigma(\tau)$, the source-biased summary of the target database. Note that the updating process requires the simple updating of the term-frequency statistics in the source-biased summary based on the statistics extracted from the new batch of sampled documents. This process repeats until a stopping condition is met. Figure 89 illustrates the source-biased probing process.

In general, we may extend the pairwise source-biased probing algorithm along two dimensions: to consider k bias sources with one target database, or to consider k target databases with one source.

Let $\mathcal{S}$ be a subset of databases from the universe of discourse $\mathcal{U}$. $\mathcal{S}$ consists of k ($0 \leq k \leq d$) databases from $\mathcal{U}$, where $\mathcal{S} = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$. Each $\sigma_j \in \mathcal{S}$ corresponds to a database $D_i \in \mathcal{U}$. We may then define a set of biased summary estimates for σ_j based on the sources of bias in $\mathcal{S}$: $\{\text{ESUMMARY}_{\sigma_1}(\tau), \text{ESUMMARY}_{\sigma_2}(\tau), ...\text{ESUMMARY}_{\sigma_k}(\tau)\}$. Hence,

with source-biased probing, a target database may be viewed through the lens of a set of biasing sources, rather than through a single unbiased summary. As a result, we may use these biased summaries to characterize the target database, which can be helpful for clustering and categorization applications, as well as for supporting query routing to the appropriate Deep Web database. For example, we may discover that the PubMed-biased summary of Monster contains many more medical-related job postings than a PubMed-biased summary of the technical jobs site Dice. Hence, a query engine could choose to route medical-jobs-related queries to Monster, rather than Dice.

Similarly, a set of target databases can be evaluated and compared with respect to one source using the source-biased lens. Extending the pairwise source-biased probing algorithm along this dimension allows for the evaluation and ranking of the target databases with respect to the source database. We discuss this type of analysis in great detail in the following section.

The performance and effectiveness of the source-biased probing algorithm depends upon a number of factors, including the network characteristics of the target database (like uptime, latency, etc.), the ranking algorithm used at the target database for ranking the returned documents, the selection criterion used for choosing source-specific candidate probe terms, and the type of stop condition used to terminate the probing process. In this chapter, we focus our attention on the selection criterion and the stop probing condition.

Mechanisms to Select Probe Terms

There are several possible ways to select the probes based on the statistics stored with each resource summary, including uniform random selection and selection based on top-weighted terms. In general, the selection criterion will recommend a query term drawn from the set of all non-zero weighted terms in the unbiased source summary ESUMMARY(σ).

Uniform Random Selection: In this simplest of selection techniques, each term that occurs in ESUMMARY(σ) has an equal probability of being selected, i.e., $Prob(\text{selecting term } j) = \frac{1}{N_\sigma}$.

Weight-Based Selection: Rather than randomly selecting query terms, we could instead rely on a ranking of the terms by one of the statistics that are recorded with each resource summary. For example, all terms in ESUMMARY(σ) could be ranked according to the weight

of each term. Terms would then be selected in descending order of weight. Depending on the type of weight cataloged (e.g., $dbFreq$, $docCount$, etc.), several flavors of weight-based selection may be considered.

Different Types of Stop Probing Conditions

The stop probing condition is the second critical component in the source-biased probing algorithm. It is interesting to note that the more documents the source-biased probing algorithm extracts from a target database, the more likely the source-biased estimated summary of the target database will tend to closely correlate with the unbiased estimated summary of the target database, meaning that the choice of stop probing condition is vitally important. So long as there are still query terms available in the source summary for use in probing, the source-biased probing algorithm will continue to extract documents, even if the queries issued are less indicative of the source subject matter. For example, if we are using PubMed as a source of bias for probing a non-relevant sports database, a poorly selected stop probing condition could result in the algorithm exhausting all of the heavily-weighted scientific and medical query probes, forcing the algorithm to send lowly-weighted (and hence, less indicative of PubMed) probes, resulting in a target summary that closely resembles an unbiased summary. Hence, the stop probing condition is critical to guide the quality of the source-biased probing algorithm. We consider four different types of conditions that might be used in practice:

Number of Queries: After some fixed number of query probes ($MaxProbes$), end the probing. This condition is indifferent to the number of documents that are examined for each database.

Documents Returned: In contrast to the first technique, the second condition considers not the number of queries, but the total number of documents ($MaxDocs$) returned by the database.

Document Thresholding: Rather than treating each document the same, this third alternative applies a threshold value to each document to determine if it should be counted toward $MaxDocs$. For each document, we may calculate the relevance of the document to the source of bias ESUMMARY(σ). If the document relevance is greater than some threshold

value, then the document is counted. Otherwise, the document is discarded.

Steady-State: Rather than relying on a count of queries or documents, this final stopping condition alternative instead relies on the estimated summary reaching a steady-state. After each probe, we calculate the difference between the new value of $\text{ESUMMARY}_\sigma(\tau)$ and the old value. If the difference (which may be calculated in a number of ways) is less than some small value ϵ, then we consider the summary stable and stop the probing.

7.4.2 Evaluating and Ranking Databases with Biased Focus

Given a source and a target database, once we generate the source-biased summary for the target database, we need an efficient mechanism to measure the source-biased relevance of a target database with respect to the source. Once a set of target databases have been evaluated with the source-biased relevance metric, we can then rank the targets with respect to the source of bias. We perform this task using the second component of source-biased database analysis – a source-biased metric.

Let σ denote a source database modeled by an unbiased summary and τ denote a target database with a σ-biased summary, and let $focus_\sigma(\tau)$ denote the source-biased focus measure. We define $focus_\sigma(\tau)$ to be a measure of the topical focus of the target database τ with respect to the source of bias σ. The focus metric ranges from 0 to 1, with lower values indicating less focus and higher values indicating more focus. In general, $focus$ is not a symmetric relation. We may describe any two Deep Web databases σ and τ with the focus in terms of σ by $focus_\sigma(\tau)$ or in terms of τ by $focus_\tau(\sigma)$. The biased focus is intended as a measure of *inclusion* [132]; that is, it measures the amount of the source included in the target.

There are several ways to calculate the biased focus for a source and a target. Adopting the cosine similarity introduced in the previous section for the case of a source-biased target summary, we may define the cosine-based focus as:

$$Cosine\_focus_\sigma(\tau) = \frac{\sum_{k=1}^{N} w_{\sigma k} w_{\tau k}^{\sigma}}{\sqrt{\sum_{k=1}^{N} \left(w_{\sigma k}\right)^2} \cdot \sqrt{\sum_{k=1}^{N} \left(w_{\tau k}^{\sigma}\right)^2}}$$

where $w_{\sigma k}$ is the weight for term k in $\text{ESUMMARY}(\sigma)$ and $w_{\tau k}^{\sigma}$ is the σ-biased weight for

term k in $\text{ESUMMARY}_\sigma(\tau)$. Again, we note that the cosine ranges from 0 to 1, with higher scores indicating a higher degree of similarity.

Alternatively, we could approximate the focus measure by computing the ratio of common terms between source and target over the source summary estimate. We call this method the *common-term based focus measure*, denoted by $CTfocus_\sigma(\tau)$.

$$CTfocus_\sigma(\tau) = \frac{|\text{ESUMMARY}(\sigma) \cap \text{ESUMMARY}_\sigma(\tau)|}{|\text{ESUMMARY}(\sigma)|}$$

This approximation counts the number of common terms between the source of bias and the target and divides by the size of the source of bias. So if all terms in the source of bias occur in the target, then the target is perfectly focused on the source and $CTfocus_\sigma(\tau) = 1$. Conversely, if no terms in the source of bias occur in the target, then the target has no focus on the source and $CTfocus_\sigma(\tau) = 0$. Unfortunately, a common-term based focus measure will tend to understate the importance of highly-weighted terms and overvalue the importance of lowly-weighted terms. An obvious solution to address the above-mentioned problem is to use the *term-weight based focus measure*, denoted by $TWfocus_\sigma(\tau)$:

$$TWfocus_\sigma(\tau) = \frac{\sum_{k \in \text{ESUMMARY}_\sigma(\tau)} w_{\sigma k}}{\sum_{k \in \text{ESUMMARY}(\sigma)} w_{\sigma k}}$$

where $w_{\sigma k}$ is the weight for term k in ESUMMARY_σ. The term weight based focus measure can be seen as a generalization of the ctf_{ratio} introduced in [37].[2]

While the $TWfocus_\sigma(\tau)$ approximation overcomes the problems of the $CTfocus_\sigma(\tau)$, it introduces new issues. For example, the term weights used in the $TWfocus_\sigma(\tau)$ approximation are from the unbiased summary of the source. Thus the actual weights of terms in the source-biased estimate may be distorted by relying on the unbiased summary weights.

Intuitively, the cosine-based biased focus is the most appealing of the three biased focus candidates since it seems to more reasonably capture the relevance between two Deep Web databases. In the experiments section we show that, compared with $TWfocus_\sigma(\tau)$ and

[2]The ctf_{ratio} is presented in the context of comparing an estimated resource summary DB' to an actual resource summary DB. $ctf_{ratio} = \sum_{i \in DB'} ctf_i / \sum_{i \in DB} ctf_i$, where ctf_i = number of times term i occurs in the source. Here, we have generalized this formulation for comparison of summaries from different databases, and for use with term weightings other than the ctf.

$CTfocus_{\sigma}(\tau)$, the $Cosine\_focus_{\sigma}(\tau)$ measure can quickly approximate the actual focus measure using fewer documents.

Ranking Relevant Databases

Given an appropriate biased focus measure, we may probe a group of target databases to identify the most relevant databases to the source of bias. For a single source of bias D_1 from our universe of discourse $\mathcal{U}$, we may evaluate multiple target databases D_2, D_3, ..., D_d. For each target database, we may evaluate the appropriate focus measure for each source-target pair (i.e., $focus_{D_1}(D_2)$, $focus_{D_1}(D_3)$, etc.). We may then rank the target databases in descending order in terms of their source-biased focus with respect to D_1.

7.4.3 Identifying Interesting Inter-database Relationships

The critical third component of source-biased database analysis are the techniques for exploiting and understanding relationships between Deep Web databases using a source-biased lens. By analyzing the nature of the relationships between Deep Web databases, we will provide support for relationship-centric queries. For example, we may identify relationship sets for a source that support queries of the form: "What other Deep Web databases are most similar to X? Or complementary to X?", among others.

As we have discussed before, current search and directory technologies for comparing Deep Web databases (such as search engines or Yahoo!-like directories) are not optimized for the type of relationship-centric queries that have a strong source-biased flavor. Some examples were given in the introduction of this chapter. In contrast, our source-biased probing framework and biased focus measure provide the flexible building blocks for automated identification of interesting relationships between Deep Web databases, especially since the framework promotes an asymmetric source-biased view for any two Deep Web databases. Our relationship discovery module creates a flexible organization of Deep Web databases, where each database is annotated with a list of relationship sets. The two typical relationship types we have identified are similarity-based and hierarchical-based.

Similarity-Based Relationships

Given the universe of discourse $\mathcal{U} = \{D_1, D_2, \ldots, D_d\}$, we identify three similarity-based

relationship sets for a particular Deep Web database D_i. These relationship sets are defined in terms of threshold values λ_{high} and λ_{low}, where $0 \leq \lambda_{low} \leq \lambda_{high} < 1$.

$\lambda -$ **equivalent**: The first relationship says that if both $focus_{D_i}(D_j) > \lambda_{high}$ and $focus_{D_j}(D_i) > \lambda_{high}$ hold, then we may conclude that D_i is sufficiently focused on D_j and D_j is sufficiently focused on D_i. Hence, the two databases are approximately the same in terms of their content coverage. We call this approximate equality λ-equivalence. It indicates that the equivalence is not absolute but is a function of the parameter λ_{high}. Formally:

$$\lambda\text{-}equivalent(D_i) = \{\forall D_j \in \mathcal{U} \mid focus_{D_i}(D_j) > \lambda_{high} \wedge focus_{D_j}(D_i) > \lambda_{high}\}$$

$$\lambda\text{-}equivalent(D_i) = \{\forall D_j \in \mathcal{U} \mid focus_{D_i}(D_j) > \lambda_{high} \wedge focus_{D_j}(D_i) > \lambda_{high}\}$$

$\lambda -$ **mutex**: If both $focus_{D_i}(D_j) < \lambda_{low}$ and $focus_{D_j}(D_i) < \lambda_{low}$ hold, then we can conclude that D_i and D_j are sufficiently concerned with different topics since neither one is very focused on the other. We annotate this approximately mutually exclusive (mutex) nature with the λ prefix. Formally:

$$\lambda\text{-}mutex(D_i) = \{\forall D_j \in \mathcal{U} \mid focus_{D_i}(D_j) < \lambda_{low} \wedge focus_{D_j}(D_i) < \lambda_{low}\}$$

$\lambda -$ **overlap**: When two Deep Web databases D_i and D_j are neither λ-equivalent nor λ-mutex, we say that the two Deep Web databases λ-overlap. Formally:

$$\lambda\text{-}overlap(D_i) = \{\forall D_j \in \mathcal{U} \mid D_j \notin \lambda\text{-}mutex(D_i) \wedge D_j \notin \lambda\text{-}equivalent(D_i)\}$$

Hierarchical Relationships

In addition to similarity-based relationship sets, we also define hierarchical relationship sets by measuring the relative coverage of target databases in $\mathcal{U}$ with respect to a particular Deep Web database D_i (source). These hierarchical relationship sets are defined in terms of a parameter λ_{diff}, where $0 \leq \lambda_{diff} \leq 1$.

$\lambda -$ **superset**: If $focus_{D_i}(D_j) - focus_{D_j}(D_i) > \lambda_{diff}$, then a relatively significant portion of D_i is contained in D_j, indicating that D_j has a λ-superset relationship with D_i. We use

the λ prefix to indicate that D_j is not a strict superset of D_i, but rather that the relationship is parameterized by λ_{diff}. Formally:

$$\lambda\text{-}superset(D_i) = \{\forall D_j \in \mathcal{U} \mid focus_{D_i}(D_j) - focus_{D_j}(D_i) > \lambda_{diff}\}$$

$\lambda - \textbf{subset}$: Conversely, If $focus_{D_j}(D_i) - focus_{D_i}(D_j) > \lambda_{diff}$, then a relatively significant portion of D_j is contained in D_i, indicating that D_j has a λ-subset relationship with D_i. Similarly, D_j is not a strict subset of D_i, but rather the relationship is parameterized by λ_{diff}. Formally:

$$\lambda\text{-}subset(D_i) = \{\forall D_j \in \mathcal{U} \mid focus_{D_j}(D_i) - focus_{D_i}(D_j) > \lambda_{diff}\}$$

We note that the determination of the appropriate λ-values is critical for the correct assignation of databases to each relationship set. In our experiments section, we illustrate how these relationship sets may be created; for now, we leave the optimization of λ-values as future work.

Using Relationship Sets

Both similarity-based and hierarchy-based inter-database relationships can be generated automatically, and used as metadata annotation to each of the Deep Web databases. These source-biased relevance data provide a flexible foundation for relationship analysis among Deep Web databases. For any Deep Web database D_i, we need only consult the appropriate relationship set to evaluate a relationship-centric query. The three similarity-based relationship sets provide the basis for answering queries of the form: "What other databases are most like X? Somewhat like X? Or complementary to X?". The two hierarchical-based sets provide the basis for answering queries of the form: "What other databases are more general than X? Or more specialized than X?". Of course, the relationship-centric queries may be further refined by considering a number of criteria besides topical relevance, including trust, quality-of-service, and size, among others.

In addition, these relationship sets are useful for routing regular document-centric queries to appropriate databases. For example, a user interested in medical literature may

choose to query both PubMed and all of the databases that have a λ-*equivalence* relationship with PubMed. Alternatively, a user interested in maximizing coverage of multiple topically-distinct Deep Web databases, may choose to query both the source database she knows about and any members in the mutually exclusive set of the source database. The hierarchical relationship sets are particularly helpful in cases where a user may refine a query to more specialized resources, or alternatively, may choose to generalize the scope of the query by considering databases further up the hierarchy to get more matching answers. In this chapter, our goal is to illustrate the importance of relationship sets and show that they may be discovered using source-biased probing. We anticipate the further exploration of relationship sets in our future work.

7.5 Focal Term Probing

One of the critical parameters to the success of source-biased probing is the choice of probe terms from the source of bias σ. We have discussed several selection techniques as well as different ways to define stop-probing conditions. In this section we introduce a refinement over these simple selection techniques whereby the source summary is segmented into k groups of co-occurring terms. The main idea is to iteratively select one term from each of the k groups to probe the target. We call these terms the focal terms of the corresponding group. When used in conjunction with the general source-biased probing algorithm, we have an enhanced version called *source-biased probing with focal terms*. Like the basic algorithm of source-biased probing, the goal remains to produce source-biased target resource summaries that are effective for detecting interesting relationships between a source of bias and a target. A unique advantage of using focal terms is that these source-biased summaries of target databases can be generated in far fewer queries and with higher quality.

7.5.1 Focal Terms and Focal Term Groups

Let σ denote a source database with its unbiased resource summary ESUMMARY_σ. We denote the set of terms with non-zero weight in ESUMMARY_σ (i.e., the terms that actually occur in the database σ) as $Terms(\sigma)$, where $Terms(\sigma)$ consists of n terms t_1, t_2, ..., t_n. A *focal term group* is a subset of terms in the set $Terms(\sigma)$ that co-occur in the documents of

Table 16: Example Focal Terms for PubMed

Group	Terms
1	care, education, family, management, ...
2	brain, gene, protein, nucleotide, ...
3	clinical, noteworthy, taxonomy, ...
4	experimental, molecular, therapy, ...
5	aids, evidence, research, winter, ...

σ. We denote a focal term group i as $FTerms_i$. The main idea behind source-biased probing with focal terms is to partition the set $Terms(\sigma)$ into k disjoint term groups such that the terms within each term group co-occur in documents of σ more frequently than they do with terms from other term groups. We note this measure of co-occurrence is rather coarse; our notion of co-occurrence merely indicates that two words occur in the same document together, regardless of their semantic relationship. Recall that in the vector-space model adopted in this chapter, the order of words within a document and the nearness of one word to another in a document are not considered, though we anticipate refining the measure of co-occurrence in future work.

Formally, we need an algorithm that can find a partition of $Terms(\sigma)$ into k focal term groups:

$$Terms(\sigma) = \{FTerms_1, \ldots, FTerms_i, \ldots, FTerms_k | \bigcup_{i=1}^{k} FTerms_i = \{t_1, ..., t_n\} \text{ and}$$

$$FTerms_i \cap FTerms_j = \emptyset\}$$

In Table 16, we show an example of five focal term groups for a collection of 100 PubMed documents. Note that k is intended to be very small since the focal term groups are meant to be very coarse. We will describe the concrete algorithm to find k partitions of the set $Terms(\sigma)$ in the next section.

Given k focal term groups, by selecting a focal term from each term group $FTerms_i$ as a probing query, we hope to retrieve documents that also contain many of the other words in that focal term group. For example, suppose we are using a frequency-based measure for query probe selection from PubMed. The top four query terms may be “brain”, “gene”,

"protein", and "nucleotide". Suppose these four terms tend to co-occur with each other as indicated in Table 16. By sending the first query "brain" to a target database, we could reasonably expect to find the other three terms since our analysis of the source indicates that these four terms tend to co-occur. A naive source-biased prober would ignore this co-occurrence information and, instead, send the other three queries "gene", "protein", and "nucleotide", even though we might reasonably expect for those queries to generate documents similar to the first query "brain". In essence, we will have used four queries when a single query would have sufficed at adequately exploring the term space of the target. In cases in which both the source and target database have similar term co-occurrences, then we would anticipate focal term probing providing an advantage over the other probe selection techniques.

The sophistication of source-biased probing with focal terms is to identify these co-occurrence relationships in order to reduce the number of queries necessary to efficiently detect relationships between a source and a target database. By using focal terms, we may generate more accurate biased summaries of target databases in far fewer probe queries and with higher quality.

In an ideal case, every focal term group would consist of terms that only co-occur with each other and not with any other terms in the other focal terms groups. By selecting a single term from each perfectly segmented term group, we ideally could send no more than k probes, one for each focal term group. Each probe would produce a document that contained every other term in that focal term group. In the more realistic setting, we will need to handle varying degrees of co-occurrence, but we still expect a good reduction in the number of probes necessary to generate a high-quality biased summary estimate for each target database.

It is important to note that, unlike previous research in grouping terms – for query-expansion [144, 185] or finding similar terms [158] – our goal is not to find close semantic relationships between terms, but rather to find very coarse co-occurrence associations among terms to support a more efficient and effective biased resource summary estimation. For example, though we may discover that "brain" and "protein" tend to co-occur, we do not

claim that there is a close semantic relationship between the two terms.

7.5.2 Finding Focal Terms

Now that we have discussed the motivation of finding focal terms, we are still faced with the task of actually segmenting $Terms(\sigma)$ into k groups of focal terms. In this section, we discuss how we may adapt a popular clustering technique to the problem of focal term discovery. Recall that in Section 7.3.1, we view a Deep Web database D_i as a set of documents, each of which is described by a vector of terms and weights. We now invert our view of a database using the same set of information. We consider a database D_i as a collection of *terms*, each of which is described by a vector of the documents in which the term occurs and a weight describing the occurrence frequency of the term in the corresponding document. Hence, we have: $Terms(D_i) = \{term_1, term_2, \cdots, term_N\}$.

For the N terms in the database, each $term_j$ $(1 \leq j \leq N)$ is a vector of documents and weights:

$$term_j = \{(doc_1, w_{j1}), (doc_2, w_{j2}), \cdots, (doc_M, w_{jM})\}$$

We can define a segmentation technique for finding focal term groups by clustering the set $Terms(D_i)$ into k clusters. Given the term vectors and the similarity function, a number of clustering algorithms can be applied to partition the set $Terms(D_i)$ of N terms into k clusters. We choose Simple K-Means since it is conceptually simple and computationally efficient. The algorithm starts by generating k random cluster centers. Each term is assigned to the cluster with the most similar (or least distant) center. The similarity is computed based on the closeness of the term and each of the cluster centers. Then the algorithm refines the k cluster centers based on the centroid of each cluster. Terms are then re-assigned to the cluster with the most similar center. The cycle of calculating centroids and assigning terms in $Terms(D_i)$ to k clusters repeats until the cluster centroids stabilize. Let C denote a cluster in the form of a set of terms in the cluster. The centroid of cluster C is:

FocalTerms(Number of Clusters k, Input Vectors $\mathcal{D}$)
Let $\mathcal{D} = \{d_1, ..., d_n\}$ denote the set of n term vectors
Let M denote the total number of documents in $\mathcal{D}$
Let $d_j = < (doc_1, w_j1), \ldots, (doc_M, w_{jM}) >$ denote a term vector of M elements,
w_{jl} is the TFIDF weight of the doc_l in term j $(l = 1, \ldots, M)$
Let $\mathcal{C} = \{C_1, ..., C_k\}$ denote a clustering of $\mathcal{D}$ into k clusters.
Let μ_i denote the center of cluster C_i
foreach cluster C_i
Randomly pick a term vector, say d_j from $\mathcal{D}$
Initialize a cluster center $\mu_i = d_j$, where $d_j \in \mathcal{D}$
repeat
foreach input term vector $d_j \in \mathcal{D}$
foreach cluster $C_i \in \mathcal{C}$ $i = 1, \ldots, k$
compute $\delta_i = sim(d_j, \mu_i)$
if δ_h is the smallest among $\delta_1, \delta_2, \ldots, \delta_k$
μ_h is the nearest cluster center to d_j
Assign d_j to the cluster C_h
// refine cluster centers using centroids
foreach cluster $C_i \in \mathcal{C}$
foreach doc l in d_j $(l = 1, \ldots, M))$
$cw_{ij} \leftarrow \frac{1}{|C_i|} \sum_{l=1}^{M} w_{jl}$
$\mu_i \leftarrow < (doc_1, cw_{i1}), \ldots, (doc_M, cw_{iM}) >$
until cluster centers no longer change
return $\mathcal{C}$

Figure 90: Focal Term Clustering Algorithm

$$centroid_C = \left\{ \begin{array}{c} (doc_1, \frac{1}{|C|} \sum\limits_{j \in C} w_{j1}) \\ (doc_2, \frac{1}{|C|} \sum\limits_{j \in C} w_{j2}) \\ \cdots \\ (doc_M, \frac{1}{|C|} \sum\limits_{j \in C} w_{jM}) \end{array} \right\}$$

where w_{jl} is the weight of term j in document l, and the formula $\frac{1}{|C|} \sum_{l \in C} w_{jl}$ denotes the average weight of the document l in the cluster C. A sketch of the K-Means term clustering based on term-vector of a Deep Web database is provided in Figure 90.

The similarity function used in Figure 90 can be defined using a number of functions. In this chapter, we use the cosine similarity function. Given a set of N terms and a set of M documents, where w_{ik} denotes the weight for term k in document i $(1 \leq k \leq N,$

$1 \leq i \leq M$), the cosine function prescribes:

$$sim(term_i, term_j) = \frac{\sum_{k=1}^{N} w_{ik} w_{jk}}{\sqrt{\sum_{k=1}^{N} (w_{ik})^2} \cdot \sqrt{\sum_{k=1}^{N} (w_{jk})^2}}$$

In Section 7.7 we report experiments on the effectiveness of using focal terms to optimize the source-biased probing algorithm, showing that the source-biased algorithm with focal terms results in more efficient probing for varying numbers of focal-term groups.

7.5.3 Selecting Focal-Based Probes

Once the k focal term groups have been constructed for a source, the remaining problem is how to select the best terms for probing a target database. We rely on a simple round-robin selection technique whereby a single term is selected from each focal term group in turn. In each round, a single term may be selected according to one of the probe selection techniques discussed above, like uniform selection or weighted selection. Once a single term has been selected from each group, the cycle repeats by selecting a second term from each group, a third term, and so on. The cycle of selecting focal terms and querying the target database may be stopped according to one of the stop probing conditions discussed above. Given this basic strategy, we may use a number of techniques for determining the order by which to select terms from the k groups and for selecting probe terms from each focal term group. One way to determine the order of focal term groups is based upon the size of each group. We begin with the group with the most terms and end each cycle with the group that has the smallest number of terms.

7.6 Implementing Source-Biased Database Analysis in DynaBot

In this section, we present the preliminary design and architecture of the DynaBot system for supporting relationship-centric queries over the Deep Web. The Deep Web of online Web-enabled databases is a large and growing component of the Web – with recent estimates suggesting that there are nearly 92,000 terabytes of data on the Deep Web versus only 167 terabytes on the surface Web [117]. Traditional crawling and indexing techniques that have shown tremendous success on the surface Web of hyperlinked pages are insufficient for the

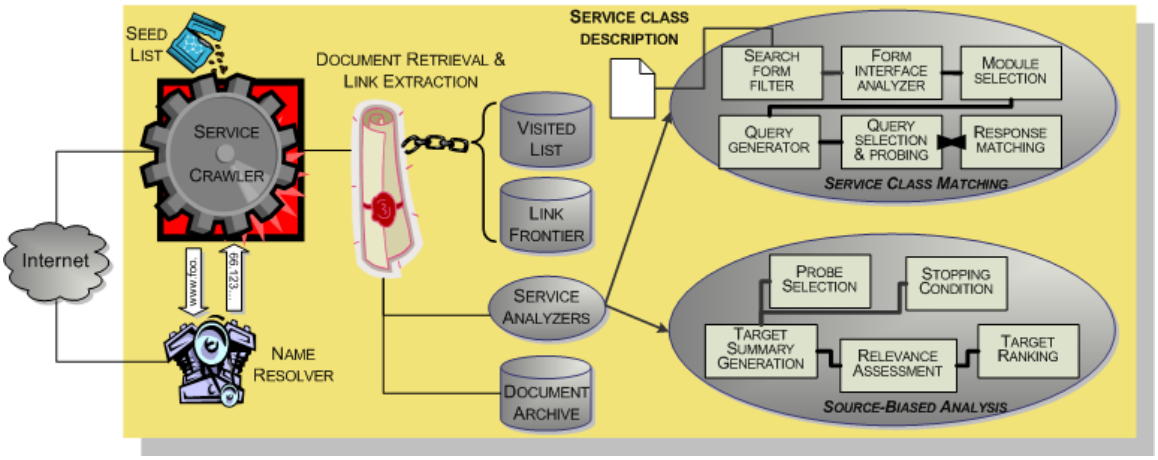

Figure 91: DynaBot System Architecture

Deep Web – where data is stored in databases or produced in real-time in response to a particular user query.

In response to these challenges, we have developed the DynaBot system for harnessing the vast amounts of data available in Web-enabled databases [150]. DynaBot is designed with a modular architecture to support the entire lifecycle of supporting Deep Web data access, ranking, and query support – including the relationship-centric queries introduced in this chapter. Figure 91 presents the overall architecture of DynaBot, with an emphasis on the initial crawling, discovery, and source-biased analysis modules.

DynaBot utilizes an advanced crawler architecture that includes standard crawler components like a URL frontier manager, network interaction modules, global storage and associated data managers, and document processors, as well as the pluggable DynaBot-specific semantic analyzers, which analyze the candidate Deep Web databases. We note that the name resolver in Figure 91 takes a URL and converts it into the corresponding IP address. The current semantic analyzers that have been incorporated into DynaBot include the *service class matcher* and the *source-biased analyzer*.

Rather than rely on a generic crawling strategy, we use the focused crawling framework to guide DynaBot on domain-specific crawls for identifying Deep Web databases related to a specific topic. Focused crawling has previously been introduced for guiding traditional Web crawlers to Web pages related to a specific topic [38]. A DynaBot crawl begins from

a seed list of URLs that may be supplied by the user and geared toward a particular domain. So a DynaBot crawl for PubMed-related resources could begin from a URL seed list including PubMed and URLs that point to PubMed. The *service class matcher* uses focused crawling of the Deep Web to discover candidate Deep Web databases that are relevant to a specific domain of interest – e.g., bioinformatics sources, online retailers, etc. The main idea of the service class matcher is to support guided matching of candidate Deep Web databases by finding members of a common *service class* of functionally similar Deep Web databases. Each service class is encoded in a service class description that describes the key attributes that define the service class. When the crawler comes across a Web-based query interface, it invokes the service class matcher to determine if it is indeed a candidate of the service class. The service class matcher includes tools for probing a candidate Deep Web site, generating a site-specific script for interacting with a Deep Web database, and classifying a site as a member of a particular service class based on the service class description. The classification component considers the input schema of the Deep Web database's query interface, the output schema of sampled data, and a seed list of example probing templates for the particular service class. The output of the *service class matcher* is a set of functionally similar Deep Web databases for use by the *source-biased analyzer*. For the PubMed example, the Deep Web databases identified would all be members of a service class for medical and scientific literature databases. The DynaBot crawling module may be run for multiple service class instances to discover a large and diverse set of Deep Web databases for consideration by the subsequent modules.

The *source-biased analyzer* module uses the site-specific scripts generated in the service class matching module for interacting with each Deep Web database discovered. The source-biased probing, biased focus evaluation, and relationship-set discovery introduced in this chapter are all incorporated into the source-biased analyzer module of DynaBot. Our current efforts are focused on continuing to enhance the capability and efficiency of these two modules, as well as incorporating additional semantic analyzers for enhanced Deep Web discovery.

7.7 Experiments

In this section, we describe five sets of experiments designed to evaluate the benefits and costs of our source-biased approach compared to existing approaches. The first set of experiments intends to show the effectiveness of our source-biased probing algorithm and performance comparison with query probing and unbiased probing. The second set evaluates the biased focus measure for ranking Deep Web database. The third set is designed to show the efficiency of the biased focus measure in identifying interesting inter-database relationships. The fourth set of experiments evaluates the efficacy of source-biased probing with focal terms by comparing the basic source-biased probing versus source-biased probing with varying number of groups of focal terms. Our experiments show that source-biased probing with focal terms can achieve about ten percent performance improvement over the basic algorithm for source-biased probing. And the final set of experiments considers the impact of several key parameters on the overall performance of the source-biased approach.

We choose two different sets of Deep Web databases for our experiments: (1) a large collection of newsgroups designed to emulate the diversity and scope of real-world Deep Web databases; and (2) a modest collection of real-world Deep Web Deep Web databases. Since the contents of Deep Web databases in the Deep Web collection change frequently and are beyond our control, and in an effort not to overload any one site, we relied on the newsgroup dataset for rigorous experimental validation. We additionally note that a similar newsgroup setup has been used before to emulate Deep Web databases [93].

Newsgroup Collection: We collected articles from 1,000 randomly selected usenet newsgroups over the period June to July 2003. We eliminated overly small newsgroups containing fewer than 100 articles, heavily spammed newsgroups (which have a disproportionate number of off-topic messages), and newsgroups with primarily binary data. After filtering out these groups, we were left with 590 *single topic* newsgroups, ranging in size from 100 to 16,000 articles. In an effort to match the heterogeneity and scope inherent in many real-world Deep Web databases, we constructed 135 additional groups of *mixed topics* by randomly selecting articles from anywhere from 4 to 80 single topic newsgroups, and 55 *aggregate topic* newsgroups by combining articles from related newsgroups (e.g., by selecting

random documents from all the subgroups in comp.unix.* into a single aggregate group). In total, the newsgroup collection consists of over 2.5GB worth of articles in 780 groups.

Deep Web Collection: For the second collection, we randomly selected 50 sites from the ProFusion [3] directory of Deep Web sites, in addition to Google and PubMed. We queried each site with a randomized set of single-word probes drawn from the standard Unix dictionary, and collected a maximum of 50 documents per site. Previous research has indicated that small samples of several hundred pages may result in high quality resource summaries over databases consisting of over one million pages, since the distribution of terms across the documents of a text database follows a Zipfian distribution [36]. In this case, we choose a sample size of 50 documents to determine if even smaller samples can yield positive results in the source-biased context.

Probing Framework: We built a probing engine in Java 1.4 for use in all of our experiments. For each group in both datasets, we constructed the actual resource summary based on the overall term frequency of each term ($dbFreq$). We eliminated a set of common stopwords (e.g., "a", "the", and so on) as well as collection-specific stopwords (e.g., "wrote", "said", and so on for the newsgroup collection). Terms were not stemmed.

7.7.1 Effectiveness of Source-Biased Probing

The goal of our first set of experiments is to compare source-biased probing with existing probing techniques such as query probing and unbiased probing and to evaluate the efficiency and quality of source-biased probing. The source-biased probing shows significant gain in terms of the percentage of documents probed that are similar to the source. For this experiment, we assume that the document download costs outweigh the query issuing cost. Hence we evaluate the efficiency of source-biased probing in terms of the number of documents required to be extracted from each target and the percentage of the documents extracted that are similar to the source. The higher percentage of documents similar (relevant) to the source, the more effective a probing algorithm is.

We selected 100 random source-target pairs from the newsgroup collection. For each

[3] `http://www.profusion.com/`

pair, we evaluated four probing techniques – a source-biased prober (*Source Bias*) that selects probe terms from the source summary in decreasing order of $dbFreq$; a query-biased prober (*Query Bias 1*) that randomly selects probes from the standard Unix dictionary of English terms; a query-biased prober (*Query Bias 2*) that selects its initial probe from the Unix dictionary, but once the first document has been retrieved from the target, all subsequent probes are selected based on the estimated $dbFreq$ of the target's resource summary; and an unbiased prober (*No Bias*) that selects documents at random from each target. For each pair, we evaluated each of the four probing techniques for up to 100 total documents extracted from each target, collecting a maximum of 5 documents per probe query from each target.

In Figure 92, we show the average percentage of documents similar (relevant) to the source ($Cosine\_focus_{\sigma}(\tau)$) over all 100 source-target pairs as a function of the number of documents examined in each target. The percentage of the documents extracted that are similar to the source (biased $focus$ measure) indicates the quality of document being extracted from each target. We see that the source-biased probing outperforms the *No Bias* prober and the *Query Bias 1* prober, resulting in an average source similarity that is initially 35% higher down to 13% after 100 documents have been extracted. Similarly, the source-biased prober outperforms the *Query Bias 2* prober, resulting in an average source similarity that is initially 57% higher down to 18% after 100 documents. Clearly, the higher focus value means the higher success for a probing algorithm.

Figure 93 shows another experiment where we also identified, in our set of 100 source-target pairs, all of those pairs that were *a priori* similar (e.g., `comp.sys.mac.apps` and `comp.sys.mac.system`) or dissimilar (e.g., `rec.crafts.textiles.sewing` and `comp.lang.perl.misc`). We show the relative performance of the *Source Bias*, *Query Bias 1*, and *No Bias* probers against these similar and dissimilar pairs. The *Query Bias 2* results track closely with the *Query Bias 1* results, and we drop them from this figure. The source-biased prober requires fewer documents to achieve the same relevance level as the other probers for all 100 source-target pairs and for the similar and dissimilar pairs. For example, for the similar source-target pairs in Figure 93, the source-biased prober identifies target documents with

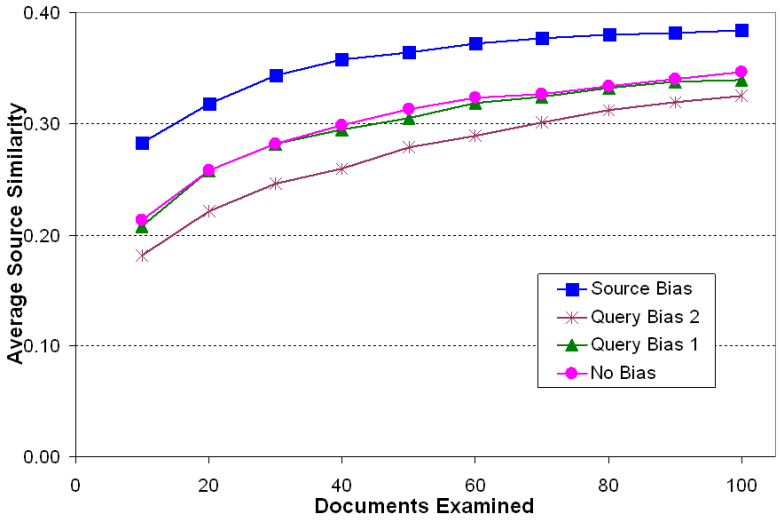

Figure 92: Probing Efficiency for 100 Source-Target Pairs

0.8 focus after extracting only 30 documents. In contrast, the other probers require between two and three times as many documents to achieve the same quality.

The third experiment is shown in Figure 94. Here we want to show how quickly a source-biased prober can hone on the most source-relevant documents in a target by plotting the percentage of the documents extracted that are similar (relevant) to the source for each of the four probers. As shown in Figure 94, the source-biased prober performs nearly two-times better than other probers: over 70% of the first 10 documents extracted from a target are source-relevant, whereas the other probers identify between 25% and 45% source-relevant documents. As more documents are examined for each target, the source-biased prober continues to maintain an advantage over the other probers. Since the source-biased prober extracts the highest-quality source-related documents from the target database first, we see the gradual decline in the *Source Bias* line, meaning that it performs the best in extracting relevant documents. We see fluctuations in the other approaches due to the randomness inherent in the query selection process. Unlike the source-biased prober which sends its best queries first, the other query probers may send a high-quality (source-relevant) query at any point of the querying process (or not at all), leading to the fluctuations in the quality of the extracted documents.

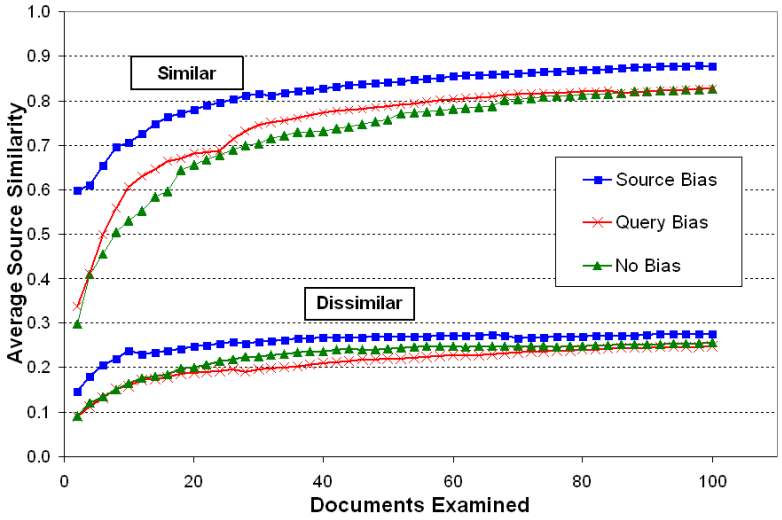

Figure 93: Probing Efficiency for Similar and Dissimilar Pairs

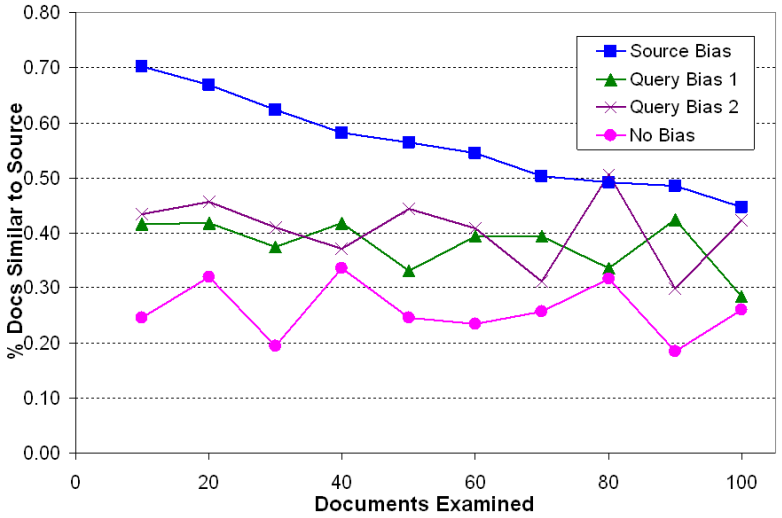

Figure 94: Average Document Quality for 100 Pairs

Table 17: Identifying Databases Relevant to PubMed

Query Bias	Source Bias
1. **AMA**	1. **Open Directory (13)**
2. **WebMD**	2. **Google (27)**
3. Linux Journal	3. **About (11)**
4. **HealthAtoZ**	4. **WebMD (2)**
5. DevGuru	5. **AMA (1)**
6. FamilyTree Magazine	6. **HealthAtoZ (4)**
7. **Mayo Clinic**	7. **Monster (22)**
8. Novell Support	8. **Mayo Clinic (7)**
9. Random House	9. Random House (9)
10. January Magazine	10. BBC News (12)

7.7.2 Ranking Effectiveness with Biased Focus

The second set of experiments intends to evaluate how well source-biased probing compares with the alternative techniques when it comes to evaluating a collection of target databases. We performed two experiments using the Deep Web collection. In the first, we use PubMed as the source and examine all 50 Deep Web databases as targets. We computed the biased focus score using $Cosine\_focus_{\sigma}(\tau)$ and then ranked all target databases relative to PubMed using the biased focus measure. Since the Deep Web sites do not support random document selection, we are unable to evaluate an unbiased prober. So this experiment only compares the source-biased prober with query biased prober 1. Table 17 shows the top-10 ranked sites relative to PubMed. In the *Source Bias* column we also list in parenthesis the rank of each site assigned by the *Query Bias* prober.

The query-biased prober identifies several health-related sites in the Deep Web collection, but it mistakenly lists Linux Journal ahead of HealthAtoZ, as well as listing a Web development site (DevGuru) and a genealogical magazine (FamilyTree) ahead of the health-related Mayo Clinic. Overall, only four of the top-ten sites could be considered topically relevant to PubMed. In contrast, the source-biased prober's top-eight sites are somewhat relevant to PubMed. In addition to the health-related sites, the source-biased prober also identifies three general sites that offer access to medical literature (Open Directory, Google, and About) that are ranked significantly lower by the query-biased prober. Interestingly, the source-biased prober identifies a fair number of scientific and bioinformatics-related job

Table 18: Identifying Databases Relevant to Google

Query Bias	Source Bias
1. Metropolis Magazine	1. **About (3)**
2. More Business	2. **Open Directory Project (20)**
3. **About**	3. Webmonkey (7)
4. Linux Journal	4. Monster (6)
5. Family Tree Magazine	5. Metropolis Magazine (1)
6. Monster	6. Random House (12)
7. Webmonkey	7. Linux Journal (4)
8. DevGuru	8. Family Tree Magazine (5)
9. US Customs	9. HealthAtoZ (15)
10. January Magazine	10. January Magazine (10)

descriptions in the Monster jobs site, resulting in its high relevance (similarity) score to PubMed (high biased focus value). While we are encouraged by the rankings here, we note that there are some problems – for example, the three general sites (Open Directory, Google, and About) are ranked ahead of the three more medically-related sites (WebMD, AMA, and HealthAtoZ). In the next section, we show how the bilateral assessment of focus for identifying relationship sets may overcome these problems.

In the second experiment, we use Google as the source and examine all 50 Deep Web databases as targets, using the setup as described above. Table 18 shows the top-10 ranked sites relative to Google. In the *Source Bias* column we also list in parenthesis the rank of each site assigned by the *Query Bias* prober.

The query-biased prober identifies only one meaningful related site (About) in the top-10, whereas the Google source-biased prober finds both relevant sites in the top-10 – About and the Open Directory Project – and ranks them as the top-2 most relevant sites. These results are further confirmation of the impact of the source-biased approach.

As a further illustration, for the source Linux Journal, we found that the top 10 source-biased probes are: linux, Web, system, software, kernel, time, source, journal, user, file. These probes are representative of the coverage of Linux Journal and are surely more effective for discovering target databases relevant to Linux Journal than random probes. On inspection, we found that when probing the jobs site Monster, the resulting Linux-biased

Table 19: Relevance Precision for 10 Source Newsgroups

Source	No Bias	Query Bias	Source Bias
comp.unix.misc	0.1	0.0	**0.7**
gnu.emacs.help	0.1	0.3	**0.4**
rec.aviation.owning	0.1	0.2	**0.4**
rec.games.chess.misc	0.1	0.1	**0.6**
rec.org.sca	0.1	0.0	**0.4**
sci.physics.research	0.5	0.3	**0.8**
talk.religion.misc	0.1	0.1	**0.6**
soc.culture.hawaii	**0.2**	0.1	**0.2**
rec.pets.cats.misc	**0.1**	**0.1**	**0.1**
comp.sys.mac.system	**0.4**	0.0	0.1

Monster summary is skewed towards linux-related jobs and other technical jobs. In contrast, the unbiased Monster summary is less relevant to the Linux Journal since on average, Monster contains many other types of jobs besides those that are linux related.

To validate the quality of source-biased database evaluation, we next randomly selected 10 sources from the newsgroup collection to evaluate against the entire set of 780 newsgroups. We compared the three probers *Source Bias*, *Query Bias 1*, and *No Bias*. For each of the 10 sources, we measured relevance precision as the percentage of the top-10 ranked target databases that are considered relevant to the source using $Cosine\_focus_{\sigma}(\tau)$. Relevance judgments were determined by the consensus opinion of three volunteers. Note that we do not measure recall since it is so expensive to calculate, requiring a relevance judgment for each source versus *every* database in the newsgroup collection.

Table 19 shows the precision for the three probers after extracting 40 documents per target database. *Source Bias* results in the highest precision in seven of ten cases, tying with the next best prober in two cases, and losing outright in one case. For the lone failure, *Source Bias* does succeed after extracting 80 documents, indicating that the mistake may be attributable to the error inherent in probing very few documents. In general, the average precision of the source-biased prober is nearly double that of the next best prober.

In Figure 95 we show the average precision for the ten sources when increasingly more documents are extracted per target. The source-biased approach displays higher precision than both the query-biased and unbiased probers in all cases considered, especially when

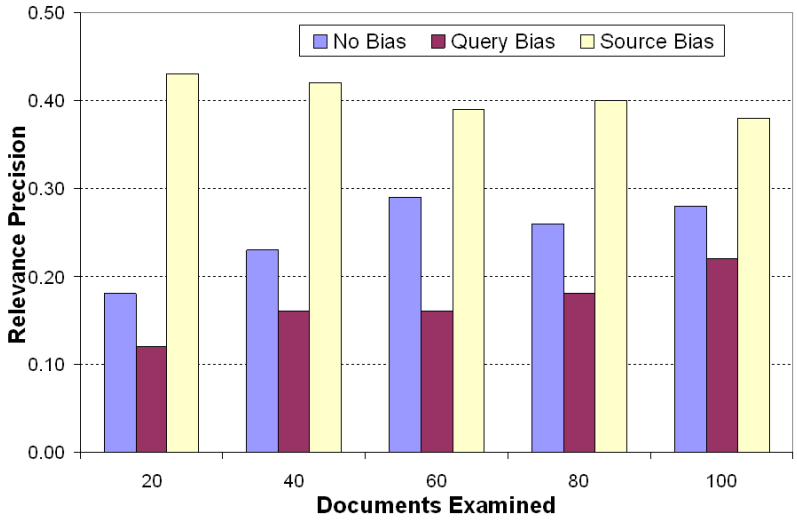

Figure 95: Average Relevance Precision

based on very few documents. We again note that the source-biased prober extracts higher quality (source-relevant) documents at the beginning of the probing, which is vitally important for limiting the amount of probing necessary to yield adequate comparisons, especially given the size and growth rate of the Deep Web. As we mentioned before, the stop probing condition is especially critical since eventually the source-biased target summary may converge to the unbiased summary as more and more documents are extracted. In this case, we can see that as the number of documents extracted increases, the two alternatives to source-biased probing improve, but still lag significantly.

While we are encouraged by the results in this section, we see that there is still some room for improvement. In the next section, we show how the bilateral assessment of focus for identifying relationship sets may yield even stronger results.

7.7.3 Identifying Inter-database Relationships

The third set of experiments is designed to evaluate the effectiveness of using the source-biased framework to support the identification of interesting inter-database relationships that the alternative schemes do not. As discussed in Section 7.4.3, the source-biased framework can identify both similarity-based relationship sets and hierarchical relationship sets for a pair of Deep Web databases or for a source database and a collection of

Table 20: Source-Biased Analysis: Identifying Relationships Relative to PubMed

Resource (D)	URL	Description	$focus_{PM}(D)$	$focus_D(PM)$	Relationship
WebMD	www.webmd.com	Health/Medical	0.23	0.18	λ-equivalent
AMA	www.ama-assn.org	Health/Medical	0.19	0.16	λ-equivalent
HealthAtoZ	www.healthatoz.com	Health/Medical	0.18	0.16	λ-equivalent
Open Directory	dmoz.org	Web Directory	0.44	0.08	λ-superset
Google	www.google.com	Web Search Engine	0.37	0.10	λ-superset
About	www.about.com	Web Channels	0.25	0.08	λ-superset
Monster	www.monster.com	Jobs	0.14	0.08	λ-overlap
Mayo Clinic	www.mayoclinic.com	Health/Medical	0.12	0.11	λ-overlap
Silicon Investor	www.siliconinvestor.com	Finance	0.03	0.04	λ-mutex
Usenet Recipes	recipes2.alastra.com	Recipes	0.02	0.03	λ-mutex
Film Critic	www.filmcritic.com	Movies	0.01	0.03	λ-mutex

target databases. Unlike the query-biased and unbiased probers, the asymmetric nature of source-biased probing allows us to characterize the nature of the relationship beyond the single relevance ranking using biased focus measure. Identifying relationship sets requires that each database be probed once for each source of bias considered, meaning that it can be more expensive than simple query-based probing.

We first illustrate relationship sets for PubMed over the Deep Web collection. In Table 20 we show four classes of relationship sets for $\lambda_{high} = 0.15$, $\lambda_{low} = 0.05$, and $\lambda_{diff} = 0.10$ using the source-biased prober described above. In contrast to the simple relevance ranking in Table 17, we see how the source-biased framework can differentiate between the very similar resources (the λ-equivalent sites WebMD, AMA, and HealthAtoZ) and the more general resources (the λ-superset sites Open Directory, Google, and About) relative to PubMed. In addition, we can identify sites with some common content (the λ-overlap sites Monster and Mayo Clinic) and sites concerned with significantly different topics (the λ-mutex sites Silicon Investor, Usenet Recipes, and Film Critic).

Similarly, we show in Table 21 several interesting relationships derived from the newsgroup collection for $\lambda_{high} = 0.70$, $\lambda_{low} = 0.40$, and $\lambda_{diff} = 0.30$ using the *Source Bias* prober discussed before. For each source considered, we probed all databases in the newsgroup collection, evaluated the biased focus metric, and report in Table 21 the relationships discovered. Again, by relying on the source-biased database analysis we may characterize relationships sets for each source that are helpful for answering relationship-centric queries of the kind posed at the beginning of the chapter.

Table 21: Source-Biased Analysis: Identifying Relationships in the Newsgroup Collection

A	B	$focus_A(B)$	$focus_B(A)$	Relationship
comp.sys.mac.apps	comp.sys.mac.system	0.86	0.76	λ-equivalent
comp.sys.mac.system	comp.sys.mac.advocacy	0.79	0.74	λ-equivalent
sci.physics.particle	sci.physics	0.86	0.80	λ-equivalent
sci.physics.particle	mixed45	0.86	0.62	λ-subset/superset
comp.unix.misc	mixed120	0.91	0.56	λ-subset/superset
rec.boats.paddle	mixed11	0.88	0.57	λ-subset/superset
rec.sport.volleyball	rec.sport.cricket	0.47	0.46	λ-overlap
rec.games.go	rec.games.chess.misc	0.50	0.53	λ-overlap
comp.os.linux	comp.unix	0.43	0.48	λ-overlap
rec.crafts.textiles.sewing	comp.lang.perl.misc	0.35	0.32	λ-mutex
comp.sys.mac.system	misc.immigration.usa	0.23	0.36	λ-mutex
comp.lang.c++	talk.religion.misc	0.21	0.29	λ-mutex

As an example, we identify `sci.physics.particle` as a member of the λ-subset relationship set of the mixed topic newsgroup `mixed11`, which consists of 25% physics-related articles in addition to articles on backgammon, juggling, and telecommunications. Interestingly, we can see that there are several overlapping relationships between newsgroups in related but slightly different fields (e.g., the two sports newsgroups `rec.sport.volleyball` and `rec.sport.cricket` and the game-related newsgroups `rec.games.go` and `rec.games.chess.misc`). Finally, we also identify several unrelated newsgroups, including `comp.sys. mac.system` relative to `misc.immigration.usa` and `comp.lang.c++` relative to `talk.religion.misc`.

7.7.4 Probing with Focal Terms

In our fourth set of experiments, we consider the impact of focal term probing on the success rate of source-biased probing. We evaluate four flavors of focal term probing – with the number of focal term groups k from which to draw source-biased probes set to 2, 3, 5, and 10. In our experiments with focal term probing, we discovered that there was little impact on either the efficiency of probing or the quality of target database evaluation when considering sources from the single-topic newsgroup collection.

In contrast, we discovered that focal term probing had a significant impact when used on mixed topic newsgroups, in which there are documents from several unrelated single topic newsgroups. In Figure 96, we show the probing efficiency for the four focal term source-biased probers relative to the best basic source-biased prober for 10 source-target pairs from the newsgroup collection. In each case, the sources were drawn exclusively from

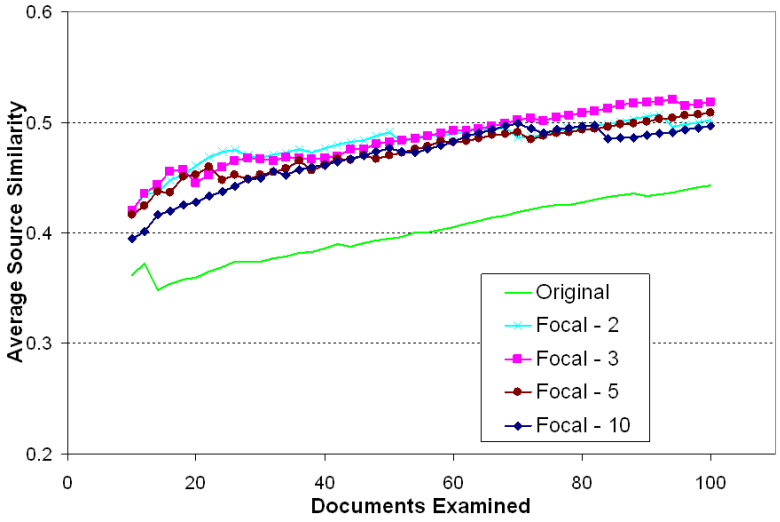

Figure 96: Impact of Focal Term Probing

the mixed topic newsgroups.

All of the focal term techniques resulted in more efficient probing versus basic source-biased probing and only minor differences in ranking precision and relationship set generation quality, indicating that focal term probing can be advantageous in certain circumstances. Our intuition is that identifying focal terms is considerably more important in cases in which there are clear distinctions in term distributions as would be reflected in the mixed topic newsgroups in which several groups of documents are concerned with different topics.

7.7.5 Varying Key Parameters

For our final set of experiments, we evaluate the impact of several key parameters on the efficiency of source-biased probing. Again, we selected 10 sources and 10 targets at random from the entire newsgroup collection, resulting in 100 source-target pairs. The first parameter we consider is the choice of query selection for source-biased probing. We consider three alternatives: random probe selection (*Source Bias (random)*), probe selection based on the overall frequency of terms in the source summary (*Source Bias (dbFreq)*), and probe selection based on the document count of each term in the source summary (*Source Bias (docCount)*). We show the results in Figure 97. The two frequency-based measures

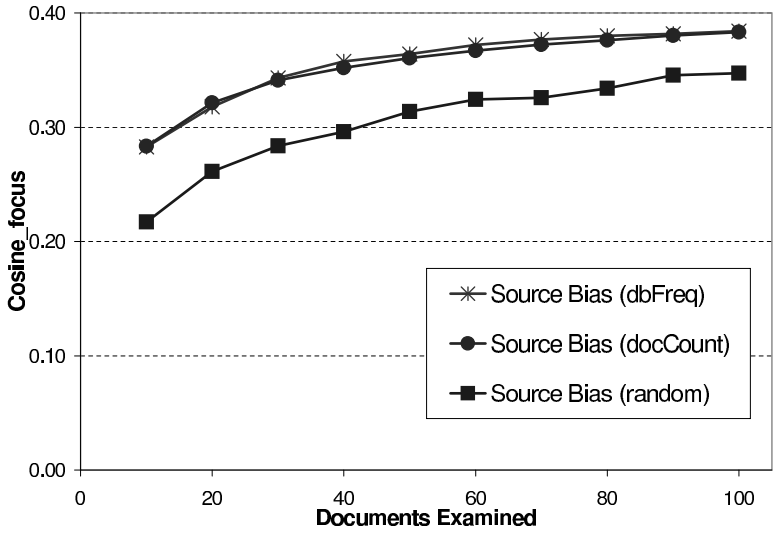

Figure 97: Query Selection Comparison

result in approximately the same quality of extracted document, requiring the extraction of fewer documents to achieve the same relevance level as the randomized source-biased prober. Since the set of candidate probes in a source's resource summary is large (on the order of 1000s), it seems reasonable to conclude that a random prober has a high likelihood of selecting non-discriminating probe terms, and hence extracting documents that are not relevant to the source of bias.

The second parameter we consider is the number of documents retrieved for each query. We considered the *Source Bias* prober described above, but we now vary the number of documents we retrieve per query, from 5 up to 20. As you can see in Figure 98, there is little change, so varying retrieved documents appears not to have a significant impact on the quality of source-biased probing.

The third parameter we compare is the choice of focus measure. In Figure 99, we compare the three versions of focus first discussed in Section 7.4.2: $Cosine\_focus_{\sigma}(\tau)$, $TWfocus_{\sigma}(\tau)$, and $CTfocus_{\sigma}(\tau)$. The dashed lines indicate the actual value of the overall focus measures calculated based on the actual resource summaries of the sources and targets. The upper dashed line corresponds to the actual $TWfocus_{\sigma}(\tau)$. The other two dashed lines correspond to $Cosine\_focus_{\sigma}(\tau)$ and $CTfocus_{\sigma}(\tau)$ and are overlapping. The first critical point to note is that both the $TWfocus_{\sigma}(\tau)$ and $CTfocus_{\sigma}(\tau)$ are slow to

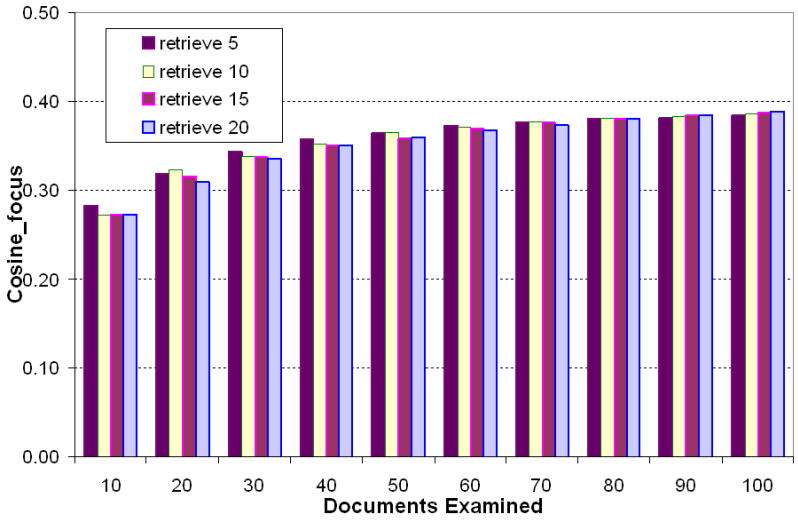

Figure 98: Documents Retrieved Comparison

approach the actual overall focus as indicated by the dashed lines, indicating that substantially more documents must be extracted per target before the estimated focus can be considered reliable. In contrast, the cosine-based focus approaches the actual focus in only 45 documents on average, further bolstering our claims for its use. Additionally, we note that $Cosine\_focus_{\sigma}(\tau)$ slightly overestimates the actual focus. This is reasonable, since for source-biased estimates based on very few documents, we would expect to identify high-quality documents first. Hence, the focus should be an overestimate. As the number of documents examined in a target nears the total available in the target, this upward bias should disappear.

One of the critical parameters to the overall success of source-biased probing with respect to comparing a source and a target is the quality of the original source summary. In this set of experiments, we investigate the degree to which the source summary quality impacts the effectiveness of source-biased probing. We again randomly selected 5 sources to compare to the entire set of 780 groups. For each source D_i $(1 \leq i \leq 5)$, we constructed three resource summaries by extracting either 100%, 10%, or 1% of the documents using the *No Bias* prober. A resource summary based on 100% of the documents is exactly the actual summary $\text{ASUMMARY}(D_i)$. The resource summaries based on 10% and 1% of the total documents are estimated summaries of decreasing quality. For each of the 5 sources, we

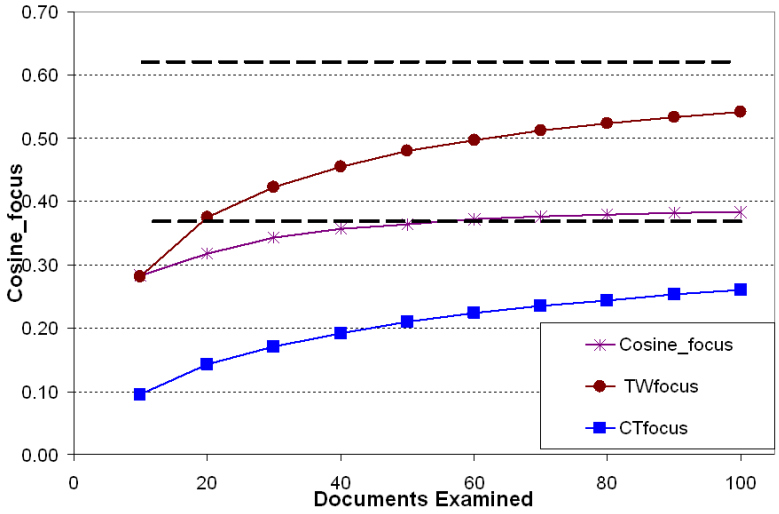

Figure 99: Comparison of Three Focus Measures

identified the number of relevant groups in the entire dataset (call this total r for each source). For non-obvious cases, we determined relevance by the consensus opinion of three volunteers. We then evaluated the summary quality by collecting 20 documents for each candidate target and ranking the targets by the $Cosine\_focus_{\sigma}(\tau)$ metric. We calculated the effectiveness for each source as the percentage of relevant targets ranked in the top-r. In Figure 100, we show the relevance ranking effectiveness for each source for each of the three resource summaries. The overall relevance is not impacted very much by the degradation in the quality of the resource summary. In three of the cases, the relevance either remains the same or falls slightly as the source summary quality decreases in quality. In two of the cases tested, the relevance precision increases slightly as the source summary quality decreases in quality. We attribute this phenomenon to the randomness inherent in the target summary probing, and aim to study it further in future work.

In our final experiment, we further illustrate how source summaries based on fairly small data samples may perform nearly as well as the actual source summaries for evaluating a target database. In Figure 101, we show the impact of the source summary quality for one source-target pair (including additional data for a 50% summary and a 5% summary). Interestingly, the relative slope for each curve is approximately the same, with only the 1% summary shifted down significantly from the performance of the actual (100%) summary.

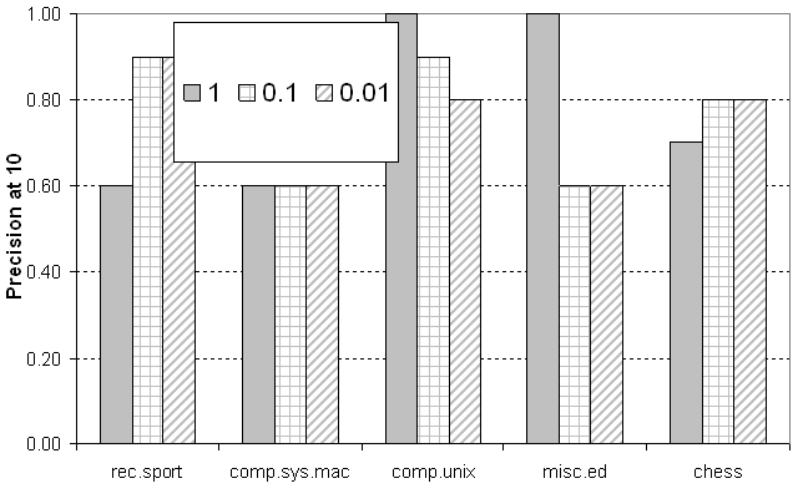

Figure 100: Impact of Source Summary Quality on Ranking

This suggests that relying on fairly small source samples (e.g., 5%) may be reasonable for evaluating Deep Web databases.

7.8 Summary

We have presented a novel source-biased approach for supporting trusted Web resource discovery. Our source-biased approach supports a relationship-centric view over a collection of Deep Web databases through source-biased probing and source-biased relevance metrics. Concretely, we have shown that the source-biased approach allows us to determine in very few interactions whether a target database is relevant to the source database by probing the target with very precise probes. The biased focus measure allows us to evaluate the relevance of Deep Web databases discovered and identify interesting types of source-biased relationships for a collection of Deep Web databases. Additionally, we have introduced source-biased probing with focal terms as a performance optimization to further improve the effectiveness of the basic source-biased model. Our experiments show that the source-biased approach outperforms query-biased probing and unbiased probing in most of the cases we have examined.

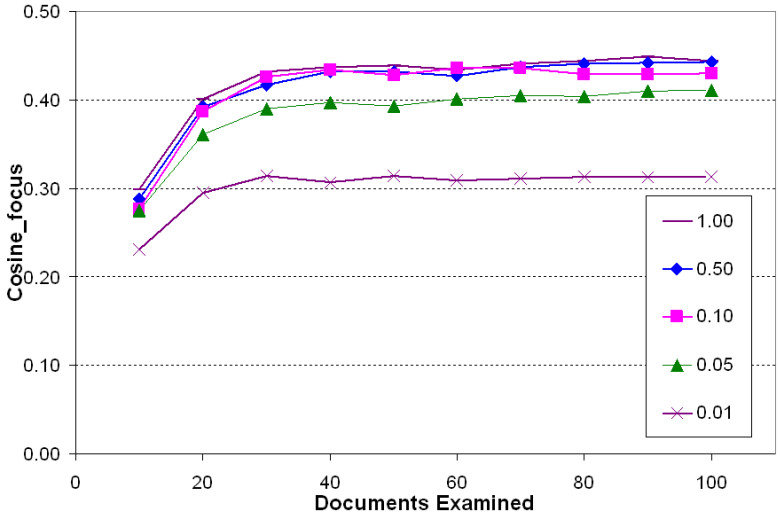

Figure 101: Impact of Resource Summary Quality

CHAPTER VIII

CONCLUSIONS AND FUTURE WORK

As the Web and Web-based open systems continue to grow and encourage far-reaching impacts – from advances in knowledge discovery to ecommerce to emerging applications – their effectiveness and quality will be under continued bombardment. As we have seen, malicious participants can undermine the quality of Web-based information and disrupt the normal functioning of Web-based systems.

In this thesis, we have developed algorithms and architectures for the efficient, reliable, and secure use of the Web where users have trust in the data and information derived from these systems, even in the presence of users intent on undermining the quality of information. We have identified three classes of vulnerabilities that threaten these systems: vulnerabilities in link-based search services, vulnerabilities in reputation-based trust services over online communities, and vulnerabilities in Web categorization and integration services. To address these risks, we have introduced a suite of methods for increasing the tamper-resilience of Web-based open systems in the face of a large and growing number of threats. Specifically, we have made new contributions in three areas:

First, we have presented a source-centric architecture and a set of techniques for providing tamper-resilient link analysis of the World Wide Web. Link-based analysis of the Web provides the basis for many important services, like Web search, Web-based data mining, and Web page categorization, and is the subject of considerable attention by spammers. We identified a number of attacks on link-based analysis of the Web - including hijacking-based attacks, honeypot-based attacks, and collusion-based attacks - that each subverts traditional link-based analysis and undermines the quality of information offered through link-based search systems. To counter these attacks, we presented a source-based link analysis approach that promotes a hierarchical abstraction of the Web graph based on the strong Web link structure in which links display strong source-based locality, for example, in terms

of administrative domains and hosts. Unlike traditional page-based link analysis, in which all pages are treated as equal nodes in a flat Web graph, this source-centric approach is promising since it captures the natural link-locality structure of the Web, supports more efficient Web applications, and naturally incorporates tamper-resilience by discounting links internal to a source. We augmented this source analysis by proposing the concept of link credibility and argueing that the intrinsic quality of a source should be distinguished from its intrinsic link credibility. We introduced a technique for assessing link credibility for all Web resources, and presented a credibility-based link analysis technique for significantly reducing the impact of malicious spammers on Web rankings. We showed that these approaches significantly reduce the impact of malicious spammers on Web rankings.

Second, we presented the design and evaluation of the SocialTrust framework for aggregating trust in online social networks. Community-based social networking systems are already extremely important and growing rapidly. For example, the popular MySpace and Facebook social networking sites boast over 100 million profiles, community-based Web discovery tools (like Digg) have 100,000s of registered users, the collaborative Wikipedia includes over 1 million articles, and eBay and Amazon rely on massive collaborative review and feedback systems. The framework supports tamper-resilient trust establishment in the presence of large-scale manipulation by malicious users, clique formation, and dishonest feedback. We have introduced relationship link quality as a scoped random walk and explored key factors impacting it. Link quality can be optimized depending on the context and application scenario and on the risk tolerance of the social network. We have developed a new random walk trust model incorporating relationship link quality and trust group feedback, and provided the first large-scale trust evaluation over real social network data. We showed that our trust models support high quality information discovery and are robust to the presence of malicious participants in the social network.

Finally, we introduced a set of techniques for reducing the opportunities of attackers to corrupt Web-based categorization and integration services. Spammers can tamper with these services by flooding them with poor quality or duplicate information sources, by advertising corrupt or untrustworthy metadata, and by infiltrating legitimate categorization

services. To mitigate the impact of such manipulation, we introduced two complementary tamper-resilient approaches for Web categorization and integration: the controlled sampling architecture and the trusted Web resource discovery system. The controlled sampling architecture supports the extraction of high-quality database samples for reducing the impact of poor quality or intentionally misleading resources. By leveraging a set of user-trusted information resources, the trusted Web resource discovery system provides a set of related information resources that can be grouped by content and trustworthiness properties to auto-generate a trusted categorization hierarchy. We showed that these techniques reduce the impact of poor quality or intentionally misleading resources and support personalized Web resource discovery.

8.1 Future Research Opportunities

Building tamper-resilient Web-based open systems is an exciting and growing area. We are interested in revisiting some of the Web risks and vulnerabilities described in Chapter 2 and in pushing this research in several interesting directions, including:

8.1.1 Incentive and Behavior Modeling

The observed attempts to manipulate Web-based open systems necessarily raise the question of *why* these manipulations occur in the first place. We believe it would be beneficial to formalize a model of behavior in a system of concern – with proper emphasis placed on the costs to each participant in terms of manipulating the system and the perceived benefits the manipulator accrues. For example, in the Web ranking manipulation studied in Chapter 3 and Chapter 4, a link-based attack incurs direct costs like domain registration fees, hosting service fees, and other maintenance costs associated with operating the spam pages; payments by colluding spammers for exchanging links; and the high cost associated with the expulsion risk for a spammer. By engaging in a Web spam attack, a spammer risks detection and potential expulsion from the ranking system, meaning a complete loss of the accumulated links and rank position. Based on a formal economic model of spam behavior, can we structure these systems to place correct economic incentives for participants to use the system properly, e.g., for the overall betterment of all participants? Can we develop

new models that are more resistant to fraud, but that still guarantee ease-of-use for the user and little disruption to the system?

8.1.2 Evolutionary Tamper-Resilience

The traditional Web data supply chain of content production, distribution, and consumption is rapidly evolving. On the content production side, more and more data is being produced by devices and sensors, and even traditional Web content is being pushed in new directions, as data is being augmented with the richer semantics associated with social media and online communities. Similarly, on the content consumption side, users are demanding content in new and innovative ways that depart from the traditional desktop mode. Mobile devices, gaming systems, and pervasive information systems embedded in our everyday life are placing new demands on content dissemination. There is a great opportunity for developing new architectures, techniques, and optimizations for enabling scalable, decentralized, and spam-resilient content delivery and distribution as the Web data supply chain continues to evolve.

REFERENCES

[1] Aberer, K. and Despotovic, Z., "Managing trust in a peer-2-peer information system," in *Proceedings of the ACM 10th Conference on Information and Knowledge Management (CIKM)*, 2001.

[2] Adali, S., Liu, T., and Ismail, M., "Optimal link bombs are uncoordinated," in *Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2005.

[3] Adamic, L. A. and Adar, E., "How to search a social network," November 2004.

[4] Agichtein, E., Ipeirotis, P., and Gravano, L., "Modeling query-based access to text databases," in *Proceedings of the Sixth International Workshop on the Web and Databases (WebDB)*, 2003.

[5] Ahrens, F., "U.S. outlaws Internet gambling," *The Washington Post*, October 2006.

[6] Amitay, E., Carmel, D., Darlow, A., Lempel, R., and So, A., "The connectivity sonar: Detecting site functionality by structural patterns," in *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia*, 2003.

[7] Androutsopoulos, I., Paliouras, G., and Michelakis, E., "Learning to filter unsolicited commercial e-mail," Tech. Rep. 2004/2, National Center for Scientific Research Demokritos, Oct. 2004.

[8] Arasu, A., Novak, J., Tomkins, A., and Tomlin, J., "PageRank computation and the structure of the web," in *Proceedings of the 11th International World Wide Web Conference (WWW)*, 2002.

[9] Attkisson, S., "MySpace being squeezed on safety," *CBS Evening News*, June 2006.

[10] Backstrom, L., Dwork, C., and Kleinberg, J., "Wherefore art thou r3579x? anonymized social networks, hidden patterns, and structural steganography," in *Proceedings of the 16th International World Wide Web Conference (WWW)*, 2007.

[11] Backstrom, L., Huttenlocher, D., Kleinberg, J., and Lan, X., "Group formation in large social networks: membership, growth, and evolution," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 2006.

[12] Baeza-Yates, R., Castillo, C., and Lopez, V., "PageRank increase under different collusion topologies," in *Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2005.

[13] Baeza-Yates, R. A. and Ribeiro-Neto, B. A., *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[14] Barnes, S. B., "A privacy paradox: Social networking in the United States," *First Monday*, vol. 11, September 2006.

[15] Benczur, A., Csalogany, K., Sarlos, T., and Uher, M., "SpamRank - fully automatic link spam detection," in *Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2005.

[16] Bergman, M., "The Deep Web: Surfacing hidden value," *BrightPlanet*, 2000.

[17] Bergstein, B., "Microsoft violates Wikipedia's sacred rule," *The Associated Press*, January 2007.

[18] Berners-Lee, T., Hall, W., Hendler, J., O'Hara, K., Shadbolt, N., and Weitzner, D. J., "A framework for web science," *Foundations and Trends in Web Science*, vol. 1, September 2006.

[19] Berners-Lee, T., Hall, W., Hendler, J., Shadbolt, N., and Weitzner, D. J., "Creating a science of the web," *Science*, vol. 313, August 2006.

[20] Beth, T., Borcherding, M., and Klein, B., "Valuation of trust in open networks," in *European Symposium on Research in Computer Security (ESORICS)*, 1994.

[21] Bharat, K. and Broder, A., "A technique for measuring the relative size and overlap of public Web search engines," in *Proceedings of the 7th International World Wide Web Conference (WWW)*, 1998.

[22] Bharat, K., Chang, B.-W., Henzinger, M. R., and Ruhl, M., "Who links to whom: Mining linkage between Web sites," in *The 2001 IEEE International Conference on Data Mining (ICDM)*, 2001.

[23] Bianchini, M., Gori, M., and Scarselli, F., "Inside PageRank," *ACM Transaction on Internet Technology (TOIT)*, vol. 5, no. 1, 2005.

[24] Boldi, P., Codenotti, B., Santini, M., and Vigna, S., "Ubicrawler: Scalability and fault-tolerance issues," in *Proceedings of the 11th International World Wide Web Conference (WWW)*, 2002.

[25] Boldi, P., Santini, M., and Vigna, S., "Do your worst to make the best: Paradoxical effects in PageRank incremental computations," in *Workshop on Algorithms and Models for the Web-Graph (WAW)*, 2004.

[26] Boldi, P. and Vigna, S., "The WebGraph Framework I," in *Proceedings of the 13th International World Wide Web Conference (WWW)*, 2004.

[27] Boyd, C., "Teenagers used to push zango on MySpace." `http://www.vitalsecurity.org/2006/07/teenagers-used-to-push-zango-on.ht%ml`, 2006.

[28] Boyd, D., "Friends, friendsters, and top 8: Writing community into being on social network sites," *First Monday*, vol. 11, December 2006.

[29] Boyd, D., "Social network sites: Public, private, or what?," *The Knowledge Tree: An e-Journal of Learning Innovation*, 2007.

[30] Broder, A., Lempel, R., Maghoul, F., and Pederson, J., "Efficient PageRank approximation via graph aggregation," in *Proceedings of the 13th International World Wide Web Conference (WWW)*, 2004.

[31] Broder, A. Z., Glassman, S. C., Manasse, M. S., and Zweig, G., "Syntactic clustering of the Web," in *Proceedings of the 6th International World Wide Web Conference (WWW)*, 1997.

[32] Cahoon, C., "Facebook phonies," *The Buchtelite (University of Akron)*, December 2005.

[33] Cai, D., He, X., Wen, J.-R., and Ma, W.-Y., "Block-level link analysis," in *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2004.

[34] Callan, J., Lu, Z., and Croft, W. B., "Searching distributed collections with inference networks," in *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1995.

[35] Callan, J., Powell, A. L., French, J. C., and Connell, M., "The effects of query-based sampling on automatic database selection algorithms," Tech. Rep. CMU-LTI-00-162, Carnegie Mellon University, 2000.

[36] Callan, J. and Connell, M., "Query-based sampling of text databases," *Information Systems*, vol. 19, no. 2, pp. 97–130, 2001.

[37] Callan, J., Connell, M., and Du, A., "Automatic discovery of language models for text databases," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1999.

[38] Chakrabarti, S., van den Berg, M., and Dom, B., "Focused crawling: a new approach to topic-specific Web resource discovery," in *Proceedings of the 8th International World Wide Web Conference (WWW)*, 1999.

[39] Chang, K. C.-C., He, B., Li, C., Patel, M., and Zhang, Z., "Structured databases on the web: Observations and implications," *SIGMOD Record*, vol. 33, no. 3, 2004.

[40] Cherry, S., "The net effect," *IEEE Spectrum*, June 2005.

[41] Cho, J., Shivakumar, N., and Garcia-Molina, H., "Finding replicated Web collections," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2000.

[42] Claburn, T., "Social networking sites feed phishers," *Information Week*, February 2007.

[43] Cohen, W. and Singer, Y., "Learning to query the Web," in *AAAI Workshop on Internet-Based Info. Systems*, 1996.

[44] comScore, "comScore releases March U.S. search engine rankings." `www.comscore.com/press/release.asp?press=1397`, April 2007.

[45] Cornelli, F., Damiani, E., , Paraboschi, S., and Samarati, P., "Choosing reputable servents in a p2p network," in *Proceedings of the 11th International World Wide Web Conference (WWW)*, 2002.

[46] Craswell, N., Bailey, P., and Hawking, D., "Server selection on the World Wide Web," in *Proceedings of the ACM Conference on Digital Libraries*, 2000.

[47] Crespo, A. and Garcia-Molina, H., "Routing indices for peer-to-peer systems," in *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, July 2002.

[48] Damiani, E., di Vimercati, S., Paraboschi, S., Samarati, P., and Violante, F., "A reputation–based approach for choosing reliable resources in peer–to–peer networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, 2002.

[49] Davison, B., "Recognizing nepotistic links on the Web," in *Workshop on AI for Web Search*, 2000.

[50] Davison, B., "Topical locality in the Web," in *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.

[51] Dhamija, R., Tygar, J. D., and Hearst, M., "Why phishing works," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, 2006.

[52] Dill, S., Kumar, R., Mccurley, K. S., Rajagopalan, S., Sivakumar, D., and Tomkins, A., "Self-similarity in the Web," *ACM Transactions on Internet Technology (TOIT)*, vol. 2, no. 3, 2002.

[53] Doan, A., Ramakrishnan, R., Chen, F., DeRose, P., Lee, Y., McCann, R., Sayyadian, M., and Shen, W., "Community information management," *IEEE Data Engineering Bulletin*, March 2006.

[54] Dodds, P. S., Muhamad, R., and Watts, D. J., "An experimental study of search in global social networks," *Science*, vol. 301, pp. 827–829, August 2003.

[55] Dolin, R., Agrawal, D., and Abbadi, A., "Scalable collection summarization and selection," in *Proceedings of the ACM Conference on Digital Libraries*, 1999.

[56] Drost, I. and Scheffer, T., "Thwarting the nigritude ultramarine: Learning to identify link spam," in *Proceedings of the 16th European Conference on Machine Learning (ECML)*, 2005.

[57] Eiron, N., McCurley, K. S., and Tomlin, J. A., "Ranking the Web frontier," in *WWW*, 2004.

[58] Ester, M., Kriegel, H.-P., and Schubert, M., "Accurate and efficient crawling for relevant websites," in *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, 2004.

[59] Fagin, R., Kumar, R., and Sivakumar, D., "Comparing top k lists," *SIAM J. Discrete Mathematics*, vol. 17, no. 1, pp. 134–160, 2003.

[60] Fetterly, D., Manasse, M., and Najork, M., "On the evolution of clusters of near-duplicate Web pages," in *Proceedings of the 1st Latin American Web Congress*, 2003.

[61] Fetterly, D., Manasse, M., and Najork, M., "Spam, damn spam, and statistics," in *Proceedings of the Seventh International Workshop on the Web and Databases (WebDB)*, 2004.

[62] Fetterly, D., Manasse, M., and Najork, M., "Detecting phrase-level duplication on the World Wide Web," in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2005.

[63] French, J. C., Powell, A. L., Callan, J. P., Viles, C. L., Emmitt, T., Prey, K. J., and Mou, Y., "Comparing the performance of database selection algorithms," in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.

[64] Frieder, L. and Zittrain, J., "Spam works: Evidence from stock touts and corresponding market activity," *Berkman Center Research Publication No. 2006-11*, March 2007.

[65] Fuhr, N., "A decision-theoretic approach to database selection in networked IR," *ACM Transactions on Information Systems (TOIS)*, vol. 17, no. 3, pp. 229–229, 1999.

[66] Gao, Y., Deng, L., Kuzmanovic, A., and Chen, Y., "Internet cache pollution attacks and countermeasures," in *Proceedings of the 14th IEEE International Conference on Network Protocols (ICNP)*, 2006.

[67] Geer, D., "Malicious bots threaten network security," *IEEE Computer*, vol. 38, no. 1, pp. 18–20, 2005.

[68] Gleich, D., Zhukov, L., and Berkhin, P., "Fast parallel PageRank: A linear system approach," tech. rep., Yahoo!, 2004.

[69] Gohring, N., "BMW cut from Google results for cheating," *PCWorld*, 6 February 2006.

[70] Goodman, J., Cormack, G. V., and Heckerman, D., "Spam and the ongoing battle for the inbox," *Communications of the ACM*, vol. 50, pp. 24–33, February 2007.

[71] Google, "Google information for webmasters." `http://www.google.com/intl/en/webmasters/`.

[72] Gravano, L. and García-Molina, H., "Generalizing GlOSS to vector-space databases and broker hierarchies," in *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB)*, 1995.

[73] Gravano, L., García-Molina, H., and Tomasic, A., "GlOSS: text-source discovery over the Internet," *ACM Transactions on Database Systems (TODS)*, vol. 24, no. 2, pp. 229–264, 1999.

[74] Gravano, L., Ipeirotis, P. G., and Sahami, M., "QProber: A system for automatic classification of hidden-web databases," *ACM TOIS*, vol. 21, no. 1, pp. 1–41, 2003.

[75] Guha, R., Kumar, R., Raghavan, P., and Tomkins, A., "Propagation of trust and distrust," in *Proceedings of the 13th International World Wide Web Conference (WWW)*, 2004.

[76] Gulli, A. and Signorini, A., "The indexable Web is more than 11.5 billion pages," in *Proceedings of the 14th International World Wide Web Conference (WWW)*, 2005.

[77] Gyöngyi, Z., Berkhin, P., Garcia-Molina, H., and Pedersen, J., "Link spam detection based on mass estimation," in *Proceedings of the 32nd International Conference on Very Large Databases (VLDB)*, 2006.

[78] Gyöngyi, Z. and Garcia-Molina, H., "Link spam alliances," in *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, 2005.

[79] Gyöngyi, Z. and Garcia-Molina, H., "Web spam taxonomy," in *Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2005.

[80] Gyöngyi, Z., Garcia-Molina, H., and Pedersen, J., "Combating Web spam with TrustRank," in *Proceedings of the 30st International Conference on Very Large Databases (VLDB)*, 2004.

[81] Harmon, A., "Amazon glitch unmasks war of reviewers," *The New York Times*, February 2004.

[82] Haugsness, K., "DNS cache poisoning detailed analysis report," *SANS Internet Storm Center http://isc.sans.org/presentations/dnspoisoning.html*, 2005.

[83] Haveliwala, T. H., "Topic-sensitive PageRank," in *Proceedings of the 11th International World Wide Web Conference (WWW)*, 2002.

[84] Hawking, D. and Thistlewaite, P., "Methods for information server selection," *ACM Transactions on Information Systems (TOIS)*, vol. 17, no. 1, pp. 40–76, 1999.

[85] Hawking, D. and Thomas, P., "Server selection methods in hybrid portal search," in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2005.

[86] He, B., Patel, M., Zhang, Z., and Chang, K. C.-C., "Accessing the Deep Web: A survey," *Communications of the ACM*, vol. 50, pp. 94–101, May 2007.

[87] He, J., Chu, W. W., and Liu, Z. V., "Inferring privacy information from social networks," in *Proceedings of the IEEE Intelligence and Security Informatics Conference*, 2006.

[88] Henzinger, M., Motwani, R., and Silverstein, C., "Challenges in Web search engines," *SIGIR Forum*, vol. 36, no. 2, 2002.

[89] IBM, "Web Services Trust Language." `http://www.ibm.com/developerworks/library/specification/ws-trust/`, 2005.

[90] Ipeirotis, P. and Gravano, L., "Improving text database selection using shrinkage," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2004.

[91] Ipeirotis, P., Gravano, L., and Sahami, M., "Probe, count, and classify: Categorizing hidden-web databases," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2001.

[92] Ipeirotis, P., Ntoulas, A., Cho, J., and Gravano, L., "Modeling and managing content changes in text databases," in *Proceedings of the 21st IEEE International Conference on Data Engineering (ICDE)*, 2005.

[93] Ipeirotis, P. G. and Gravano, L., "Distributed search over the hidden web: Hierarchical database sampling and selection," in *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB)*, 2002.

[94] Jagatic, T., Johnson, N., Jakobsson, M., and Menczer, F., "Social phishing," *Communications of the ACM*, to appear.

[95] Joachims, T., "Optimizing search engines using clickthrough data," in *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.

[96] Kamvar, S., Yang, B., and Garcia-Molina, H., "Secure score management for peer-to-peer systems," tech. rep., Stanford University, 2004.

[97] Kamvar, S. D., Haveliwala, T. H., Manning, C. D., and Golub, G. H., "Exploiting the block structure of the Web for computing PageRank," tech. rep., Stanford University, 2003.

[98] Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H., "The EigenTrust algorithm for reputation management in p2p networks," in *Proceedings of the Twelth International World Wide Web Conference (WWW)*, 2003.

[99] Kendall, M. and Gibbons, J. D., *Rank Correlation Methods*. Edward Arnold, 1990.

[100] Kleinberg, J. M., "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[101] Kotadia, M., "Samy opens new front in worm war," *CNET News.com*, October 2005.

[102] Krebs, B., "Hacked ad seen on MySpace served spyware to a million." `http://blog.washingtonpost.com/securityfix/2006/07/myspace_ad_served_ad%ware_to_mo.html`, 2006.

[103] Kumar, R., Novak, J., and Tomkins, A., "Structure and evolution of online social networks," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 2006.

[104] Lam, S. and Riedl, J., "Shilling recommender systems for fun and profit," in *Proceedings of the 13th International World Wide Web Conference (WWW)*, 2004.

[105] Langville, A. N. and Meyer, C. D., "Deeper inside PageRank," *Internet Mathematics*, vol. 1, no. 3, 2005.

[106] Lempel, R. and Moran, S., "The stochastic approach for link-structure analysis and the tkc effect," in *Proceedings of the 9th International World Wide Web Conference (WWW)*, 2000.

[107] Lempel, R. and Moran, S., "Predictive caching and prefetching of query results in search engines," in *Proceedings of the 12th International World Wide Web Conference (WWW)*, 2003.

[108] Liben-Nowell, D., Novak, J., Kumar, R., Raghavan, P., and Tomkins, A., "Geographic routing in social networks," *Proceedings of the National Academy of Sciences*, vol. 102, no. 33, pp. 11623–1162, 2005.

[109] Lin, J., "Divergence measures based on the shannon entropy," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 145–151, 1991.

[110] Liu, K.-L., Yu, C., and Meng, W., "Discovering the representative of a search engine," in *Proceedings of the ACM 10th Conference on Information and Knowledge Management (CIKM)*, 2001.

[111] Liu, L., "Query routing in large-scale digital library systems," in *Proceedings of the 15th IEEE International Conference on Data Engineering (ICDE)*, 1999.

[112] Liu, T.-Y. and Ma, W.-Y., "Webpage importance analysis using conditional markov random walk," in *Web Intelligence*, 2005.

[113] Liu, Z., Lin, W., Li, N., and Lee, D., "Detecting and filtering instant messaging spam - a global and personalized approach," in *Workshop on Secure Network Protocols (NPSEC)*, 2005.

[114] Lu, J. and Callan, J., "Federated search of text-based digital libraries hierarchical peer-to-peer networks," in *SIGIR Workshop on P2P Information Retrieval*, 2004.

[115] Lu, Y., Zhang, B., Xi, W., Chen, Z., Liu, Y., Lyu, M. R., and Ma, W.-Y., "The PowerRank Web link analysis algorithm," in *Proceedings of the 13th International World Wide Web Conference (WWW)*, 2004.

[116] Lv, Q., Cao, P., Cohen, E., Li, K., and Shenker, S., "Search and replication in unstructured peer-to-peer networks," in *Proceedings of the 16th international conference on Supercomputing (ICS)*, pp. 84–95, ACM Press, 2002.

[117] Lyman, P. and Varian, H. R., "How much information." www.sims.berkeley.edu/how-much-info-2003, 2003.

[118] Mann, C., "Spam + blogs = trouble," *Wired*, 2006.

[119] Marti, S. and Garcia-Molina, H., "Taxonomy of trust: Categorizing p2p reputation systems," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 50, pp. 472–484, March 2006.

[120] Massa, P. and Avesani, P., "Controversial users demand local trust metrics: An experimental study on epinions.com community.," in *AAAI*, 2005.

[121] Maximilien, E. M. and Singh, M. P., "Toward autonomic Web services trust and selection," in *Proceedings of the 2nd international conference on Service oriented computing (ICSOC)*, 2004.

[122] McSherry, F., "A uniform approach to accelerated PageRank computation," in *Proceedings of the 14th International World Wide Web Conference (WWW)*, 2005.

[123] Meng, W., Liu, K.-L., Yu, C. T., Wang, X., Chang, Y., and Rishe, N., "Determining text databases to search in the internet," in *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB)*, 1998.

[124] Meng, W., Yu, C. T., and Liu, K.-L., "Detection of heterogeneities in a multiple text database environment," in *Proceedings of the Fourth IFCIS International Conference on Cooperative Information Systems (CoopIS)*, 1999.

[125] Metaxas, P. T. and DeStefano, J., "Web spam, propaganda and trust," in *Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2005.

[126] Milgram, S., "The small-world problem," *Psychology Today*, pp. 60 – 67, May 1967.

[127] Mindlin, A., "Seems somebody is clicking on that spam," *The New York Times*, July 2006.

[128] Mishne, G. and Carmel, D., "Blocking blog spam with language model disagreement," in *Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2005.

[129] Monastersky, R., "The number that's devouring science," *The Chronicle of Higher Education*, October 2005.

[130] Moshchuk, A., Bragin, T., Gribble, S. D., and Levy, H. M., "A crawler-based study of spyware in the Web," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2006.

[131] Ng, A. Y., Zheng, A. X., and Jordan, M. I., "Stable algorithms for link analysis," in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.

[132] Nie, J., "An information retrieval model based on modal logic," *Information Processing & Management*, vol. 25, no. 5, pp. 477–497, 1989.

[133] Nottelmann, H. and Fuhr, N., "Evaluating different methods of estimating retrieval quality for resource selection," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2003.

[134] Ntoulas, A., Najork, M., Manasse, M., and Fetterly, D., "Detecting spam Web pages through content analysis," in *Proceedings of the 15th International World Wide Web Conference (WWW)*, 2006.

[135] Ntoulas, A., Zerfos, P., and Cho, J., "Downloading textual hidden Web content through keyword queries," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 2005.

[136] Nussbaum, E., "Kids, the Internet, and the end of privacy," *New York Magazine*, February 2007.

[137] OASIS, "OASIS Web Services Security." `www.oasis-open.org/committees/wss/`, 2006.

[138] Page, L., Brin, S., Motwani, R., and Winograd, T., "The PageRank citation ranking: Bringing order to the Web," tech. rep., Stanford University, 1998.

[139] Poitras, C., "MySpace releases names of sex offenders," *The Hartford Courant*, May 2007.

[140] Porter, M. F., "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[141] Powell, A. L., French, J. C., Callan, J. P., Connell, M. E., and Viles, C. L., "The impact of database selection on distributed searching," in *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.

[142] Provos, N., McNamee, D., Mavrommatis, P., Wang, K., and Modadugu, N., "The ghost in the browser: Analysis of Web-based malware," in *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets*, 2007.

[143] Qiu, F. and Cho, J., "Automatic identification of user interest for personalized search," in *Proceedings of the 15th International World Wide Web Conference (WWW)*, 2006.

[144] Qiu, Y. and Frei, H.-P., "Concept-based query expansion," in *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1993.

[145] Raghavan, S. and Garcia-Molina, H., "Crawling the hidden Web," in *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB)*, 2001.

[146] Ratnasamy, S. and others, "A scalable content addressable network," in *Proceedings of ACM SIGCOMM 2001*, 2001.

[147] Raynes-Goldie, K. and Fono, D., "Hyperfriendship and beyond: Friendship and social norms on Livejournal," in *Association of Internet Researchers (AOIR-6)*, 2005.

[148] Richardson, M., Agrawal, R., and Domingos, P., "Trust management for the semantic Web," in *Web, Proceedings of the Second International Semantic Web Conference*, 2003.

[149] Robertson, J., "Data theft scam targets Google ads," *The Associated Press*, April 2007.

[150] Rocco, D., Caverlee, J., Liu, L., and Critchlow, T., "Exploiting the Deep Web with DynaBot: Matching, probing, and ranking (poster)," in *Proceedings of the 14th International World Wide Web Conference (WWW)*, 2005.

[151] Rossi, G., Schwabe, D., and Guimaraes, R., "Designing personalized Web applications," in *Proceedings of the 10th International World Wide Web Conference (WWW)*, 2001.

[152] Rubin, S., Christodorescu, M., Ganapathy, V., Giffin, J. T., Kruger, L., Wang, H., and Kidd, N., "An auctioning reputation system based on anomaly detection," in *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS)*, 2005.

[153] Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E., "A bayesian approach to filtering junk e-mail," in *Proceedings of the AAAI-1998 Workshop on Learning for Text Categorization*, July 1998.

[154] Salton, G., Wong, A., and Yang, C., "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1971.

[155] Sanchez, M., "Pranksters posting fake profiles on myspace," *http://www.dfw.com/*, 2006.

[156] Sanders, H. M., "Google announces purge of ad-heavy Web sites," *New York Post*, May 2007.

[157] Scheeres, J., "Europeans outlaw net hate speech," *Wired*, November 2002.

[158] Schutze, H. and Pedersen, J. O., "A cooccurrence-based thesaurus and two applications to information retrieval," *Information Processing and Management*, vol. 33, no. 3, 1997.

[159] Seibel, J., "Boy charged in creating fake MySpace profile," *Milwaukee Journal-Sentinel*, April 2006.

[160] Si, L. and Callan, J., "Using sampled data and regression to merge search engine results," in *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.

[161] Si, L. and Callan, J., "Relevant document distribution estimation method for resource selection," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2003.

[162] Song, R., Wen, J.-R., Shi, S., Xin, G., Liu, T.-Y., Qin, T., Zheng, X., Zhang, J., Xue, G., and Ma, W.-Y., "Microsoft Research Asia at Web track and terabyte track," in *TREC*, 2004.

[163] Srivatsa, M., Xiong, L., and Liu, L., "Trustguard: Countering vulnerabilities in reputation management for decentralized overlay networks," in *Proceedings of the 14th International World Wide Web Conference (WWW)*, 2005.

[164] Stoica, I. and others, "Chord: A scalable peer-to-peer lookup service for Internet applications," in *SIGCOMM*, 2001.

[165] Stone, B., "Facebook expands into MySpace's territory," *The New York Times*, May 2007.

[166] Sugiura, A. and Etzioni, O., "Query routing for Web search engines: Architecture and experiments," in *Proceedings of the 9th International World Wide Web Conference (WWW)*, 2000.

[167] Thelwall, M., "New versions of PageRank employing alternative Web document models," *Aslib Proceedings: new information perspectives*, vol. 56, no. 1, pp. 24–33, 2004.

[168] von Ahn, L., Blum, M., Hopper, N. J., and Langford, J., "Captcha: Using hard AI problems for security," in *EUROCRYPT*, 2003.

[169] Wang, J., Wen, J.-R., Lochovsky, F., and Ma, W.-Y., "Instance-based schema matching for Web databases by domain-specific query probing," in *Proceedings of the 30th International Conference on Very Large Databases (VLDB)*, 2004.

[170] Wang, W., Meng, W., and Yu, C., "Concept hierarchy based text database categorization in a metasearch engine," in *Proceedings of the 1st International Conference on Web Information Systems Engineering (WISE)*.

[171] Wang, Y.-M., Beck, D., Jiang, X., Roussev, R., Verbowski, C., Chen, S., and King, S. T., "Automated Web patrol with Strider HoneyMonkeys: Finding Web sites that exploit browser vulnerabilities," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2006.

[172] Wang, Y.-M., Ma, M., Niu, Y., and Chen, H., "Spam double-funnel: Connecting Web spammers with advertisers," in *Proceedings of the 16th International World Wide Web Conference (WWW)*, 2007.

[173] Wang, Y. and DeWitt, D. J., "Computing PageRank in a distributed Internet search engine system," in *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, 2004.

[174] Wasserman, S. and Faust, K., *Social network analysis*. Cambridge: Cambridge University Press, 1994.

[175] Watts, D. J., "Networks, dynamics, and the small world phenomenon," *American Journal of Sociology*, vol. 105, no. 2.

[176] Watts, D. J., Dodds, P. S., and Newman, M. E. J., "Identity and search in social networks," *Science*, vol. 296, pp. 1302–1305, 2002.

[177] Wu, B. and Davison, B., "Cloaking and redirection: A preliminary study," in *Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2005.

[178] Wu, B. and Davison, B., "Identifying link farm spam pages," in *Proceedings of the 14th International World Wide Web Conference (WWW)*, 2005.

[179] Wu, B., Goel, V., and Davison, B., "Propagating trust and distrust to demote web spam," in *Models of Trust for the Web (MTW)*, 2006.

[180] Wu, B., Goel, V., and Davison, B., "Topical TrustRank: Using topicality to combat Web spam," in *Proceedings of the 15th International World Wide Web Conference (WWW)*, 2006.

[181] Wu, J. and Aberer, K., "Using SiteRank for P2P Web retrieval," tech. rep., Swiss Fed. Inst. of Tech., 2004.

[182] Wu, W., Yu, C. T., Doan, A., and Meng, W., "An interactive clustering-based approach to integrating source query interfaces on the Deep Web," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2004.

[183] Xiong, L. and Liu, L., "Supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, 2004.

[184] Xu, J. and Callan, J., "Effective retrieval with distributed collections," in *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1998.

[185] Xu, J. and Croft, W. B., "Query expansion using local and global document analysis," in *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996.

[186] Xue, G.-R., Yang, Q., Zeng, H.-J., Yu, Y., and Chen, Z., "Exploiting the hierarchical structure for link analysis," in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2005.

[187] Yang, B. and Garcia-Molina, H., "Improving search in peer-to-peer networks," in *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, July 2002.

[188] Yang, B. and Garcia-Molina, H., "Designing a super-peer network," in *IEEE International Conference on Data Engineering, 2003*, 2003.

[189] Yu, B. and Singh, M. P., "A social mechanism of reputation management in electronic communities," in *Cooperative Information Agents*, pp. 154–165, 2000.

[190] Yuan, L., Kant, K., Mohapatra, P., and Chuah, C.-N., "DoX: A peer-to-peer antidote for DNS cache poisoning attacks," in *Proceedings of the IEEE International Conference on Communications*, 2006.

[191] Yuwono, B. and Lee, D. L., "Server ranking for distributed text retrieval systems on the internet," in *Database Systems for Advanced Applications (DASFAA)*, 1997.

[192] Zhang, H., Goel, A., Govindan, R., Mason, K., and Roy, B. V., "Improving eigenvector-based reputation systems against collusions," in *Workshop on Algorithms and Models for the Web-Graph (WAW)*, 2004.

[193] Zhang, Z., He, B., and Chang, K. C.-C., "Understanding Web query interfaces: Best-effort parsing with hidden syntax," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2004.

[194] Zittrain, J. and Edelman, B., "Documentation of Internet filtering worldwide." `http://cyber.law.harvard.edu/filtering`.

VITA

James Buchanan Caverlee was born and raised in Shreveport, a city of a few hundred thousand on the banks of the Red River in northwest Louisiana. His mother Ellen was born and raised in Shreveport – in fact, James is a member of the fourth generation of his family to graduate from the same high school in Shreveport – C.E. Byrd High School. His father Sam was born in Chicago. As a child Sam migrated to Monroe, Louisiana with his mother and three brothers. Family members believe James shares many character traits with his great-grandfather Earl "Buck" Buchanan. Buck was born in 1897 in Indiana, trained as an engineer, and eventually followed the oil and gas business to Shreveport where he owned and operated the Globe Map Company. In a family of attorneys, public servants, and community volunteers strongly linked to northwest Louisiana, James and Buck share an interest in the technical, as well as a wanderlust.

Years before the advent of the Web and the ubiquitous notion of eCommerce, James managed a fledgling clicks-and-mortar operation. Weekly, he would log onto a local electronic bulletin board system to arrange comic book and baseball card transactions with other local computer aficionados. The year was 1987. His first successful computer-mediated

network-enabled purchase was X-Men Comics issue #141. In retrospect, he later realized that he had unwittingly stumbled into the intersection of technology and business, an area that would feature prominently in his scholarly development.

Years later, James majored in Economics at Duke University, where he established a base of critical thinking skills and an understanding of the core issues impacting business, ultimately graduating magna cum laude in 1996. He broadened this perspective when he subsequently worked in Washington, D.C. as a consultant for the international consulting firm LECG. With a goal of building mathematical rigor, James enrolled in the Ph.D. program at Stanford's Engineering-Economic Systems & Operations Research department, where he took courses in decision analysis, optimization, and probabilistic analysis, and earned the M.S. degree in 2000. In parallel, James consulted for the Bay Area-based strategy and consulting firms R.B. Webber and ValueScience, advising several start-up technology companies during the heady days of the Internet bubble. These experiences accentuated his keen interest in the deeper technology issues impacting the businesses he had advised, leading James to study Computer Science at Stanford (M.S. 2001) and now at Georgia Tech. Since arriving at Georgia Tech in 2002, James has been a member of the Distributed Data Intensive Systems Lab and the multidisciplinary Tennenbaum Institute for enterprise transformation. In the summer of 2004 James interned at The Mitre Corporation.

His research interests are generally in the areas of information management and distributed data-intensive systems. In particular, his research has emphasized: (1) Spam-Resilient Web-Scale Computing; (2) Web Information Retrieval and Management; and (3) Enterprise Computing and Workflow Management.

James married his college sweetheart Sherry in 2000. They welcomed a son in 2004 and a daughter in 2006.